PSP Acquire: Herramienta de soporte a la toma de datos para el Proceso de Software Personal

Proyecto Fin de Carrera

Miguel Casado Cornejo

Tutor:
Alejandro Rodríguez González

Tabla de Contenido

Tabla de Ilustraciones

Listado de tablas

Agradecimientos:

Nunca es fácil ser agradecido cuando se portan bien contigo, y mucho menos es fácil ser agradecido cuando se portan mal. Gracias a Dios ese no es mi caso, y debo dar gracias a mucha gente por lo bien que me han tratado siempre.

A mis profesores del instituto, porque ellos me exigían como a un adulto, aunque yo todavía no me estuviera preparado.

A mis profesores de la universidad, porque cuando ya me convertí en adulto no me trataban como tal, sino como a un amigo.

A mis amigos, porque siempre han estado conmigo, ya sea siendo o no adultos, y siempre se han comportado como parte de mi familia.

A mi familia, porque ellos son los que me han educado y han conseguido su mayor objetivo, convertirme en buena persona.

Muchas gracias a todos.

INTRODUCCIÓN

Introducción:

El proceso de software personal, conocido por sus siglas como PSP, es una metodología desarrollada en el Instituto de Ingeniería del Software (SEI). Se centra en aspectos como la planificación, la calidad y la estimación de costes, permitiendo aumentar la calidad de los productos de software realizados.

Por tanto podemos definir PSP (proceso de software personal) como una versión reducida de CMM (Modelo de Capacidad y Madurez). El PSP se caracteriza porque es de uso personal y se aplica a programas de menos de 10.000 líneas de código. Se centra en la administración del tiempo y de la calidad a través de la eliminación temprana de defectos. En principio, estaba dirigido a estudiantes, y a medida que fue evolucionando, empezó a dirigirse a Ingenieros principiantes. El PSP no abarca aspecto como: Trabajo en equipo, Administración de configuraciones y Administración de requerimientos.

Para lograr entender un poco más de PSP hablemos de forma superflua del modelo padre, el Modelo de Capacidad y Madurez (CMM), cuyo objetivo es la evaluación de los procesos de una organización. Este modelo establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso (KPA - Key Process Area). Para cada área de proceso define un conjunto de buenas prácticas que habrán de ser:

- Definidas en un procedimiento documentado
- Provistas (la organización) de los medios y formación necesarios
- Ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas)
- Medidas
- Verificadas

A su vez estas Áreas de Proceso se agrupan en cinco "niveles de madurez", de modo que una organización que tenga institucionalizadas todas las prácticas incluidas en un nivel y sus inferiores, se considera que ha alcanzado ese nivel de madurez.
Los niveles son:

- **Inicial**. Las organizaciones en este nivel no disponen de un ambiente estable para el desarrollo y mantenimiento de software. Aunque se utilicen técnicas correctas de ingeniería, los esfuerzos se ven minados por falta de planificación. El éxito de los proyectos se basa la mayoría de las veces en el esfuerzo personal, aunque a menudo se producen fracasos y casi siempre retrasos y sobrecostes. El resultado de los proyectos es impredecible.
- **Repetible**. En este nivel las organizaciones disponen de unas prácticas institucionalizadas de gestión de proyectos, existen unas métricas básicas y un razonable seguimiento de la calidad. La relación con subcontratistas y clientes está gestionada sistemáticamente.
- **Definido**. Además de una buena gestión de proyectos, a este nivel las organizaciones disponen de correctos procedimientos de coordinación entre grupos, formación del personal, técnicas de ingeniería más detalladas y un nivel más avanzado de métricas en los procesos. Se implementan técnicas de revisión por pares (peer reviews).
- **Gestionado**. Se caracteriza porque las organizaciones disponen de un conjunto de métricas significativas de calidad y productividad, que se usan de modo sistemático para la toma de decisiones y la gestión de riesgos. El software resultante es de alta calidad.
- **Optimizado**. La organización completa está volcada en la mejora continua de los procesos. Se hace uso intensivo de las métricas y se gestiona el proceso de innovación.

Motivación:

El desarrollo de productos de software implica mucho más que escribir instrucciones de programación y ejecutarlas en un ordenador. Requiere cumplir requisitos del cliente a un costo y planificación acordada. El PSP muestra cómo producir de forma regular software de alta calidad. Utilizando el PSP se obtienen datos que muestran la efectividad del trabajo y se identifican los puntos fuertes y las debilidades, además se practican habilidades y métodos que los ingenieros del software van a desarrollar durante muchos años de pruebas y errores.

El PSP enseña a ingenieros y futuros ingenieros, cómo administrar la calidad de sus productos y cómo hacer compromisos que ellos puedan cumplir. Puede ser empleado en muchas fases en el ciclo de desarrollo de programas pequeños, definición de requerimientos, documentación, pruebas y mantenimiento.

El diseño de PSP se basa en los siguientes principios de planeación y de calidad:

- Cada ingeniero es esencialmente diferente; es decir, los ingenieros deben planear su trabajo y basar sus planes en sus propios datos personales.
- Para mejorar constantemente su funcionamiento, los ingenieros deben utilizar personalmente procesos bien definidos y medidos.
- Para desarrollar productos de calidad, los ingenieros deben sentirse personalmente comprometidos con la calidad de sus productos.
- Para hacer un trabajo de ingeniería de software de la manera correcta, los ingenieros deben planear de la mejor manera su trabajo antes de comenzarlo y deben utilizar un proceso bien definido para realizar de la mejor manera la planeación del trabajo.
- Para que los desarrolladores lleguen a entender su funcionamiento de manera personal, deben medir el tiempo que pasan en cada proceso, los defectos que cometen y remueven de cada proyecto y finalmente medir los diferentes tamaños de los productos que llegan a producir.

Para poder cumplir los principios de PSP es necesaria la elaboración de tablas y recoger datos para su estudio.

Puesto que el mayor valor añadido de un ingeniero de software reside en el tiempo, la utilización de este método supone una pequeña pérdida de ese valor, y teniendo en cuenta que no puede actuarse en detrimento de la calidad del software final lo más apropiado es la reducción del tiempo de la toma de datos.

Por lo tanto se va a realizar un sistema que controle la recogida de datos y facilite al ingeniero de software las herramientas necesarias para medir tanto su rendimiento, como para realizar estimaciones más exactas de sus proyectos.

Alcance:

El Sistema de Gestión de PSP desarrollado en este proyecto de fin de carrera ha sido dividido en subsistemas para reducir su complejidad. El alcance del conjunto es la suma del alcance de las partes (subsistemas). La relación de subsistemas es la siguiente:

- Subsistema de Gestión de actividades. (SGA)
- Subsistema de Gestión de usuarios. (SGU)
- Subsistema de Gestión de los registros. (SGR)
- Subsistema de Registro Semanal de Actividades. (SRSA)
- Subsistema de Estimación de proyectos. (SEP)
- Subsistema de Gestión de programas. (SGP)

A continuación se procederá a describir cada uno de estos subsistemas.

Subsistema de Gestión de actividades. (SGA)

Descripción:

Este subsistema se encarga de permitir el mantenimiento de las actividades realizadas o a realizar por el usuario. Dentro de dichas actividades están incluidas las relacionadas con la realización de un proyecto y no deben ser ni modificadas ni borradas.

Beneficios:

- Permite controlar actividades ajenas a la realización de un proyecto.
- Permite un seguimiento de actividades de interés para un usuario.
- Permite priorizar actividades según el tiempo disponible.

Finalidad:

- Optimizar los tiempos de trabajo y ocio.
- Mejorar la rutina de trabajo del usuario.

Componentes:

- Alta de una nueva actividad.
- Modificación de una actividad existente.
- Borrado de una actividad existente.

Subsistema de Gestión de usuarios. (SGU)

Descripción:

Este subsistema se encarga permitir el mantenimiento de los usuarios. También se encarga de las consultas de los acumulados por parte de un usuario, así como sus valores de Valoración-fallo, LOC/Hora y rendimiento.

Beneficios:

- Permite controlar los usuarios que usan el sistema.
- Permite un seguimiento de los niveles acumulados por el usuario.

Finalidad:

- Llevar un control de los usuarios que utilizan el sistema.
- Crear una interactividad entro los distintos usuarios al ser capaces de consultar sus acumulados históricos.

Componentes:

- Alta de un nuevo usuario.
- Modificación de un usuario existente.
- Borrado de un usuario existente.
- Consulta de los datos y del acumulado de un usuario.

Subsistema de Gestión de los registros. (SGR)

Descripción:

Este subsistema se encarga permitir el mantenimiento de los registros de tiempos usados en actividades o defectos introducidos en la realización de un proyecto. Por tanto puede subdividirse en dos subsistemas:

- Subsistema de gestión de los registros de tiempo. (SGRT)
- Subsistema de gestión de los registros de defectos. (SGRD)

Subsistema de Gestión de los registros de tiempo. (SGRT)

Descripción:

Este subsistema se encarga de permitir el mantenimiento de los tiempos utilizados en cada una de las actividades.

Beneficios:

- Permite controlar el tiempo asignado a cada tarea, así como el tiempo real utilizado.
- Permite controlar el tiempo "desperdiciado" en interrupciones en las actividades.
- Permite mejorar la cantidad de tiempo útil usado en cada tarea.

Finalidad:

- Optimizar los tiempos de trabajo y ocio.
- Mejorar la rutina de trabajo del usuario.

Componentes:

- Alta de un nuevo tiempo.
- Modificación de un tiempo existente.
- Borrado de un tiempo existente.

Subsistema de Gestión de los registros de defectos. (SGRD)

Descripción:

Este subsistema se encarga de permitir el mantenimiento de los defectos introducidos durante la codificación del programa. Así como el tiempo dedicado a su eliminación y el momento en el que se produjo el defecto o su detección.

Beneficios:

- Permite controlar la localización exacta de los errores, así como el tiempo dedicado a subsanar el error.
- Permite un seguimiento de la fase de depuración, puesto que pueden existir errores no resueltos.
- Permite mejorar el nivel de programación del usuario, puesto que es necesario un mayor tiempo de búsqueda de errores.
- Permite mejorar la calidad del producto final.

Finalidad:

- Optimizar los tiempos de codificación y depuración.
- Optimizar el nivel de programación del usuario.
- Evitar defectos en el producto final.
- Mejorar la rutina de trabajo del usuario.

Componentes:

- Alta de un nuevo defecto.
- Modificación de un defecto existente.
- Borrado de un defecto existente.

Subsistema de Registro Semanal de Actividades. (SRSA)

Descripción:

Este subsistema se encarga de realizar el informe del registro semanal de actividades. Dicho informe se realiza a través de los datos introducidos por el usuario.

Beneficios:

- Valoración objetiva del trabajo.
- Motivación del usuario al ser capaz de ver una progresión o regresión en su rendimiento.

Finalidad:

- Optimizar los tiempos de codificación y depuración.
- Mejorar la rutina de trabajo del usuario.
- Motivación del usuario.

Componentes:

- Informe de Registro Semanal de Actividades.

Subsistema de Gestión de proyectos. (SGP)

Descripción:

Este subsistema se encarga de permitir el mantenimiento de los proyectos a los que se deben añadir programas. Dicha agrupación lógica permite una división temporal en el análisis de los datos, puesto que se realizará su representación divida por proyectos.

Beneficios:

- Permite estimar el tamaño del proyecto.
- A través de dicha estimación se puede ajustar una estimación del coste y de la duración del proyecto dependiendo del usuario.
- La creación de estimaciones correctas supone un ahorro para el usuario, tanto como para el cliente.

Finalidad:

- Optimizar los resultados de la estimación de duración y coste.

Componentes:

- Alta de un nuevo proyecto.
- Modificación de un proyecto existente.
- Borrado de un proyecto existente.

Subsistema de Gestión de programas. (SGP)

Descripción:

Debido a que es necesaria la subdivisión de un proyecto grande en distintos programas (módulos) este subsistema es necesario para poder gestionar dichos programas o módulos. Para cada uno de los proyectos existirá al menos un programa, por tanto es necesario gestionarlos. Cada uno de estos programas contiene unas fases propias de planificación, diseño, codificación, revisión de código, compilación, pruebas y postmortem.

Beneficios:

- Permite controlar los tiempos y costes asociados a cada programa del proyecto.
- Permite evaluar cada tipo de programa o módulo por separado, tanto su tamaño como su tasa de errores.
- Permite controlar las tareas críticas de cada programa, al estar a su vez dividido en fases.

Finalidad:

- Optimizar los resultados de la estimación de tiempo y coste.

Componentes:

- Alta de un nuevo programa.
- Modificación de un programa.
- Borrado de un programa.

Descripción general y objetivos:

La aplicación a desarrollar permitirá la introducción automática de los tiempos dedicados a cada actividad, así como el usuario que la realiza. También permitirá la gestión de los subsistemas anteriormente indicados.

El objetivo principal de la aplicación es proporcionar al usuario un sencillo método de control de PSP. Gracias a este método el usuario podrá cumplir los objetivos principales de PSP:

- Ayudar a reducir los tiempos de estimación de un proyecto, así como su coste.
- Permitir una mayor precisión al estimar dichos valores.
- Crear una rutina de trabajo que permita mejorar constantemente dichas estimaciones.
- Permitir una valoración objetiva del rendimiento del usuario.
- Gestionar las actividades realizadas por el usuario.
- Optimizar el proceso de codificación, intentando minimizar los defectos.
- Crear programas de calidad.
- Evitar retrasos en el cumplimiento de los plazos dados al cliente, así como mantener el presupuesto inicial dado al cliente.

Herramientas y tecnologías:

Hardware:

Para la realización del proyecto se va a utilizar un ordenador portátil con las siguientes características:

- Procesador: Pentium i5.
- Memoria: 4 Gb. de RAM.
- Tarjeta gráfica: 1 Gb. de RAM.
- Disco duro: 320 Gb.
- Sistema Operativo: El PFC se va a desarrollar bajo Microsoft Windows 7, por lo que implica que todo el software mencionado más adelante trabajará sobre versiones estables para este sistema operativo.

Controladores:

- No son necesarios controladores adicionales para la realización de este PFC, puesto que no son necesarios periféricos específicos.

Ingeniería de Software:

- StarUML: Programa Open Source para el desarrollo de los diagramas UML (Lenguaje de Modelado Unificado). Será utilizado para la realización de los casos de uso y los modelados de datos.
- MSProject: Programa integrado en el paquete Office de Microsoft, bajo licencia universitaria a través de MSDNAA-UC3M. Será utilizado para la realización de la planificación y estimación en costes del proyecto.

Bases de Datos:

- MS-Access: Programa integrado en el paquete Office de Microsoft. Será utilizado para la realización de la base de datos de la aplicación.

Lenguajes de programación:

- C#
- XAML
- Ambos se utilizarán bajo Microsoft Visual Studio 2010, bajo licencia universitaria a través de MSDNAA-UC3M.

Multimedia:

- Adobe Photoshop CS3. Se utilizará para la realización de las imágenes pertenecientes al proyecto, así como para la realización de las capturas de pantalla necesarias para la documentación.

Web:

- Mozilla Firefox: Navegador gratuito, se utilizará para realizar la búsqueda de información necesaria tanto como para la comunicación con el tutor del proyecto.
- Correo: Se ha creado una cuenta gratuita en Gmail para establecer una comunicación rápida con el tutor del proyecto y posibilitar el envío de entregas parciales.

Útiles:

- MS Office: Paquete ofimático de Microsoft, bajo licencia universitaria, se utilizará para la realización de diagramas sencillos, así como para la realización de la memoria.

Tutoriales:

- Se utilizará MSDN (Microsoft Developer Network) como medio de aprendizaje y consulta sobre los lenguajes de programación utilizados.

GESTIÓN DEL PROYECTO

Gestión del proyecto software:

Ciclo de vida:

Modelo Cascada

La visión del modelo cascada del desarrollo de software es el siguiente: el desarrollo de software puede ser a través de una secuencia consecutiva de fases. Cada fase tiene un conjunto de metas bien definidas, y las actividades dentro de una fase contribuyen a la satisfacción de metas de esa fase o quizás a una subsecuencia de metas de la fase. Las flechas muestran el flujo de información entre las fases. La flecha de avance muestra el flujo normal. Las flechas hacia atrás representan la retroalimentación.

El modelo de ciclo de vida cascada, captura algunos principios básicos:

- Planear un proyecto antes de embarcarse en él.
- Definir el comportamiento externo deseado del sistema antes de diseñar su arquitectura interna.
- Documentar los resultados de cada actividad.
- Diseñar un sistema antes de codificarlo.
- Testear un sistema después de construirlo.

Se elige este modelo debido a que están muy bien definidas y separadas las fases a realizar y la sencillez que supone seguir esta metodología de trabajo.

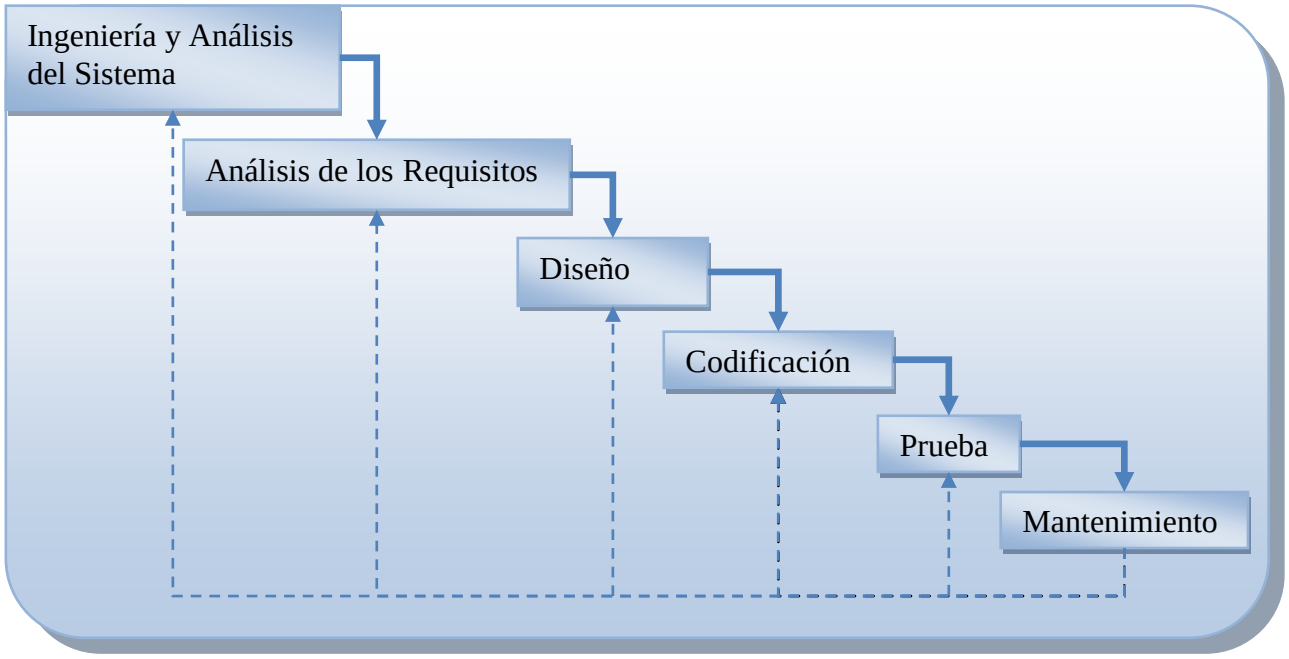


Ilustración 1: Ciclo de vida en cascada

Presupuesto:

Debido a la utilización de licencias de ámbito público como estudiante de la Universidad Carlos III de Madrid, el presupuesto queda limitado al tiempo dedicado por parte del ingeniero de software.

A dicho ingeniero de software se le asigna un coste de 30 €/hora, debido a que la mayor parte del trabajo se realiza fuera del horario laboral.

El presupuesto total desglosado por fases es el siguiente:

Introducción	930,00 €
Gestión del proyecto software	1.890,00 €
Estado del arte	380,00 €
Análisis del sistema	780,00 €
Casos de uso	1.140,00 €
Diseño del sistema	1.560,00 €
Desarrollo del Aplicativo	1.980,00 €
Implantación y explotación	630,00 €
Revisión general	360,00 €
Total	9.650,00 €
I.V.A (18%)	1.737,00 €
Total + I.V.A.	**11.387,00 €**

Tabla 1: Presupuesto total desglosado por fases

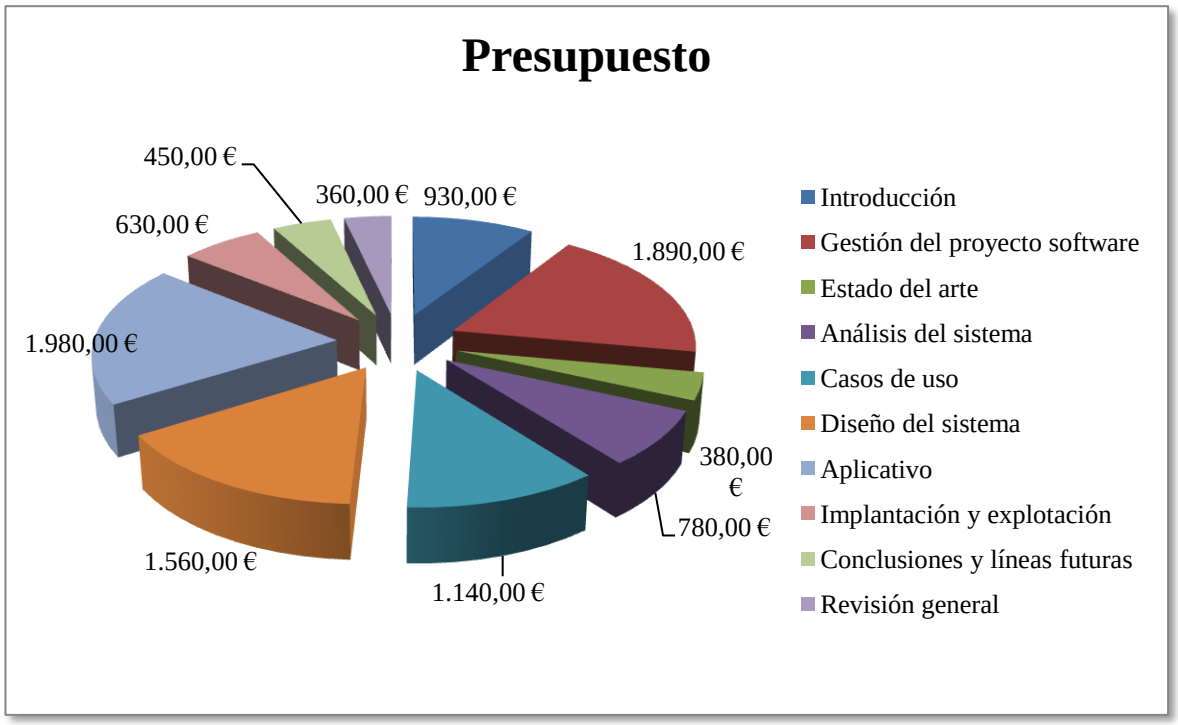


Ilustración 2: Gráfico del presupuesto

El presupuesto total por actividades es un presupuesto desglosado, definiendo las actividades que se realizan durante cada fase, y asignándolas un valor presupuestado.

Fase de Introducción:

Motivación	120,00 €
Alcance	120,00 €
Descripción General y Objetivos	270,00 €
Herramientas y Tecnologías	240,00 €
Revisión	180,00 €
Total	**930,00 €**

Tabla 2: Presupuesto Fase de Introducción

Fase de Gestión del proyecto software:

Ciclo de Vida	210,00 €
Presupuesto	330,00 €
Organización	330,00 €
Organigrama WBS	180,00 €
Organigrama RBS	270,00 €
Organigrama PBS	180,00 €
Planificación	210,00 €
Revisión	180,00 €
Total	**1.890,00 €**

Tabla 3: Presupuesto Fase de Gestión del proyecto software

Fase de Estado del arte:

Estado del arte	200,00 €
Revisión	180,00 €
Total	**380,00 €**

Tabla 4: Presupuesto Fase de Estado del arte

Fase de Análisis del sistema:

Requisitos de Capacidad	300,00 €
Requisitos de Restricción	300,00 €
Revisión	180,00 €
Total	**780,00 €**

Tabla 5: Presupuesto Fase de Análisis del sistema

Fase Casos de uso:

Especificación de Casos de Uso	210,00 €
Diagramas de Casos de Uso	300,00 €
Detalle Casos de Uso	450,00 €
Revisión	180,00 €
Total	**1.140,00 €**

Tabla 6: Presupuesto Fase Casos de uso

Fase Diseño del sistema:

Arquitectura SW	300,00 €
Diagrama de Clases	420,00 €
Modelo físico de datos	420,00 €
Esquema E/R	210,00 €
Revisión	210,00 €
Total	**1.560,00 €**

Tabla 7: Presupuesto Fase Diseño del sistema

Desarrollo del aplicativo:

Desarrollo del Aplicativo	1.800,00 €
Revisión	180,00 €
Total	**1.980,00 €**

Tabla 8: Presupuesto Fase Desarrollo del aplicativo

Fase Implantación y explotación

Manual de Instalación y explotación	180,00 €
Manual del Usuario	270,00 €
Revisión	180,00 €
Total	**630,00 €**

Tabla 9: Presupuesto Fase Implantación y explotación

Organización:

La organización utilizada para el desarrollo del proyecto se describe detalladamente en los siguientes apartados: WBS, RBS y PBS.

Organigrama WBS:

Un WBS es una presentación simple y organizada del trabajo requerido para completar el proyecto. Existen muchas maneras de organizar la presentación de este trabajo. Para la realización de este WBS se han utilizado las fases a realizar para completar la realización del proyecto.

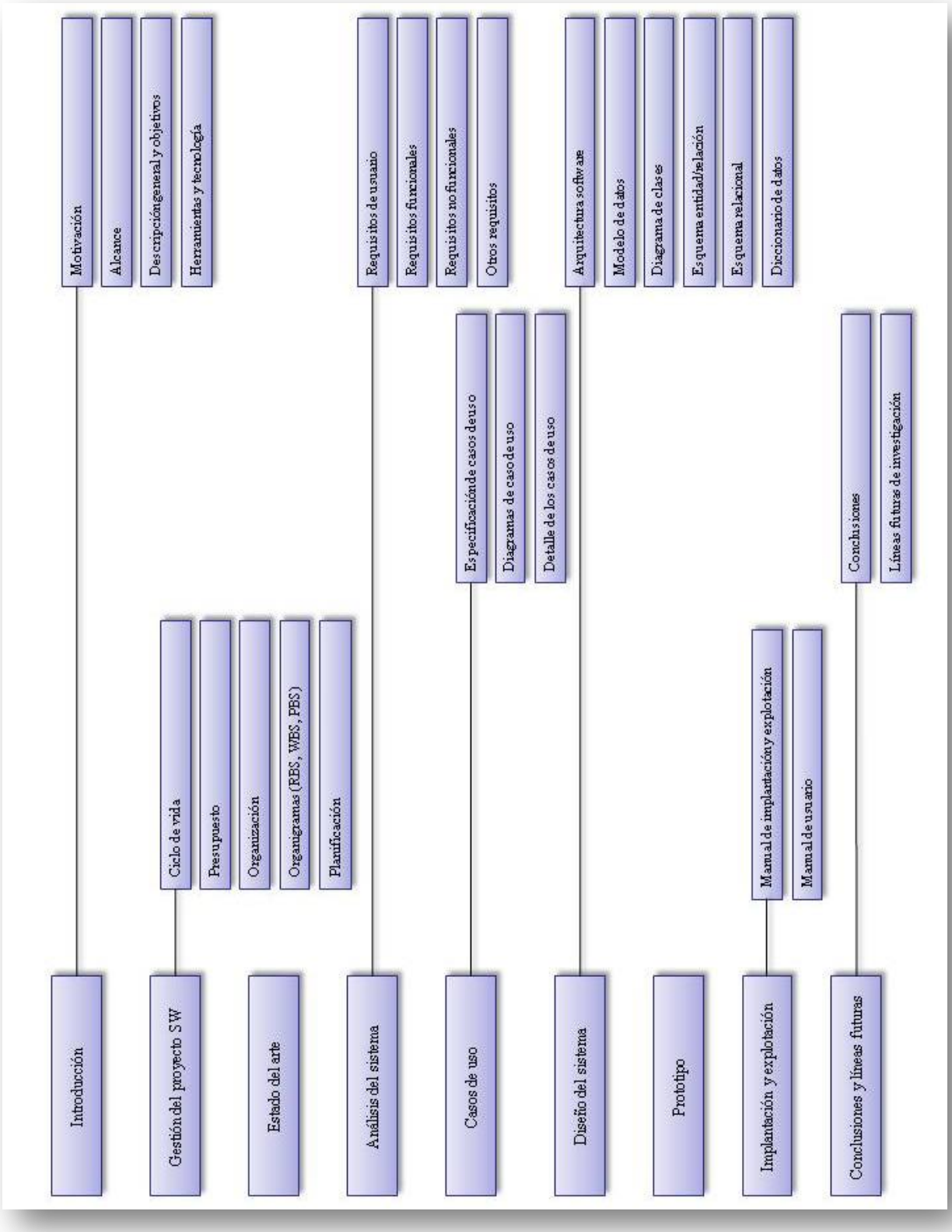


Ilustración 3: WBS

Organigrama RBS:

El RBS es un árbol jerárquico que representa los recursos humanos y materiales del proyecto. Los principales objetivos del RBS son los siguientes:

- Mostrar gráficamente la organización humana del proyecto.
- Maximizar el uso de los conocimientos y experiencia del personal disponible.
- Reflejar la estructura de recursos materiales necesarios para la realización del proyecto, así como sus costes asociados.

Para poder llevar a cabo el proyecto, se ha dispuesto de los siguientes recursos, que se antojan como imprescindibles a la hora de realizar cualquier actividad, recursos humanos y materiales.

Se empezará por definir los recursos humanos, que comprenden el grupo de trabajo y el equipo de calidad.

Recursos Humanos

El equipo de desarrollo está compuesto por un jefe de proyecto con experiencia en el desarrollo de este tipo de aplicaciones y un ingeniero informático recién licenciado.

- Jefe de Proyecto: Miguel Casado
- Analista: Miguel Casado
- Programador: Miguel Casado
- Jefe de Calidad: Miguel Casado
- Encargado de Pruebas: Miguel Casado
- Encargado de Aseguramiento de Calidad: Miguel Casado

Recursos Materiales

Ver Herramientas y Tecnologías

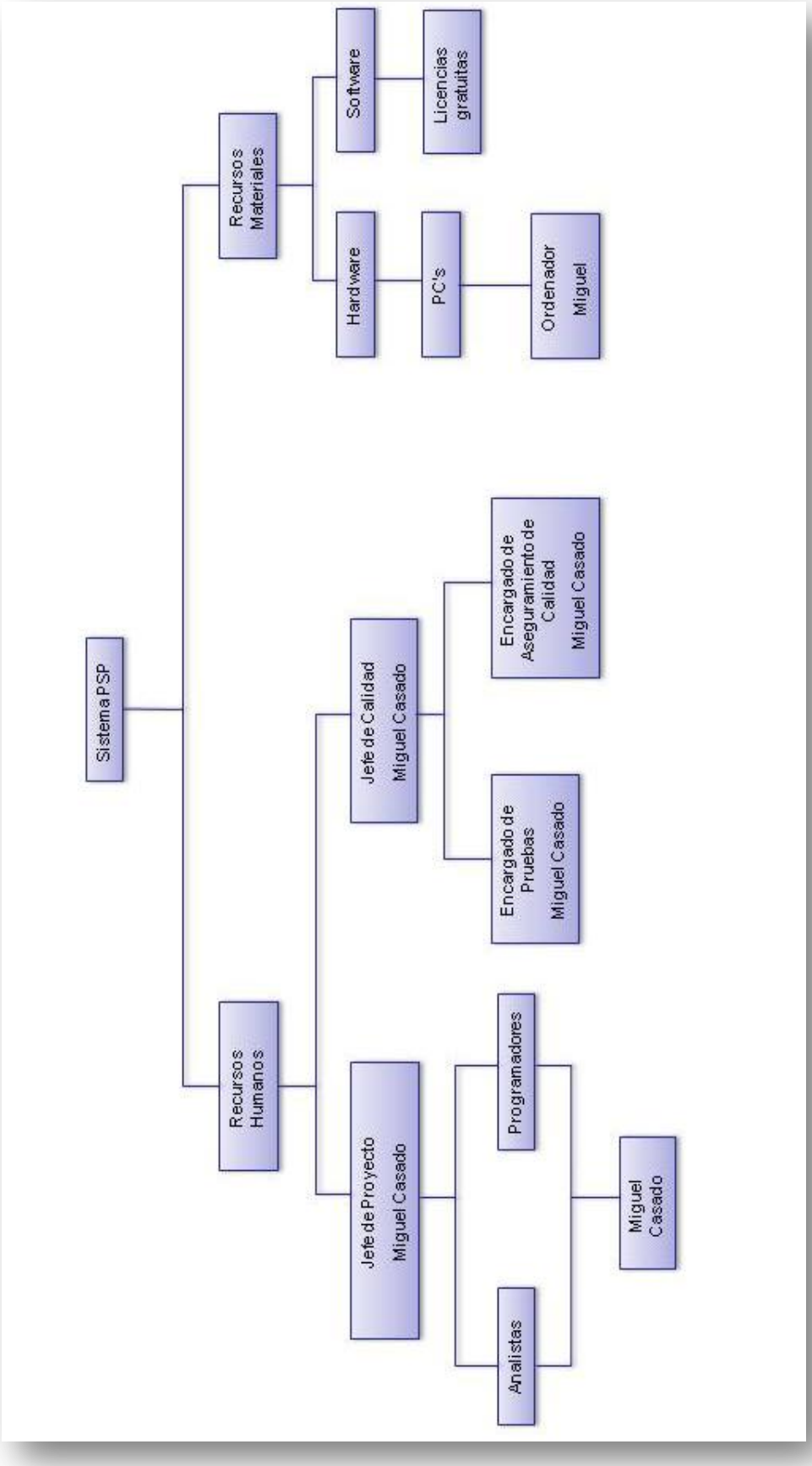


Ilustración 4: RBS

Organigrama PBS:

Esta técnica de organización tiene por objeto la representación de la estructura de partidas o elementos a entregar con el proyecto software, tanto internos como externos (de cara al Cliente o Usuario).

El PBS es una representación del producto o sistema a desarrollar en el proyecto en forma de árbol, que describe el producto a diferentes niveles, desde el más alto que representa todo el producto, y descendiendo progresivamente hacia subsistemas, elementos de configuración y componentes (nivel más bajo).

Es un método de definición de todas las partes, entregables y no entregables, del producto objeto del proyecto, que se apoya en las técnicas de gestión de la configuración.

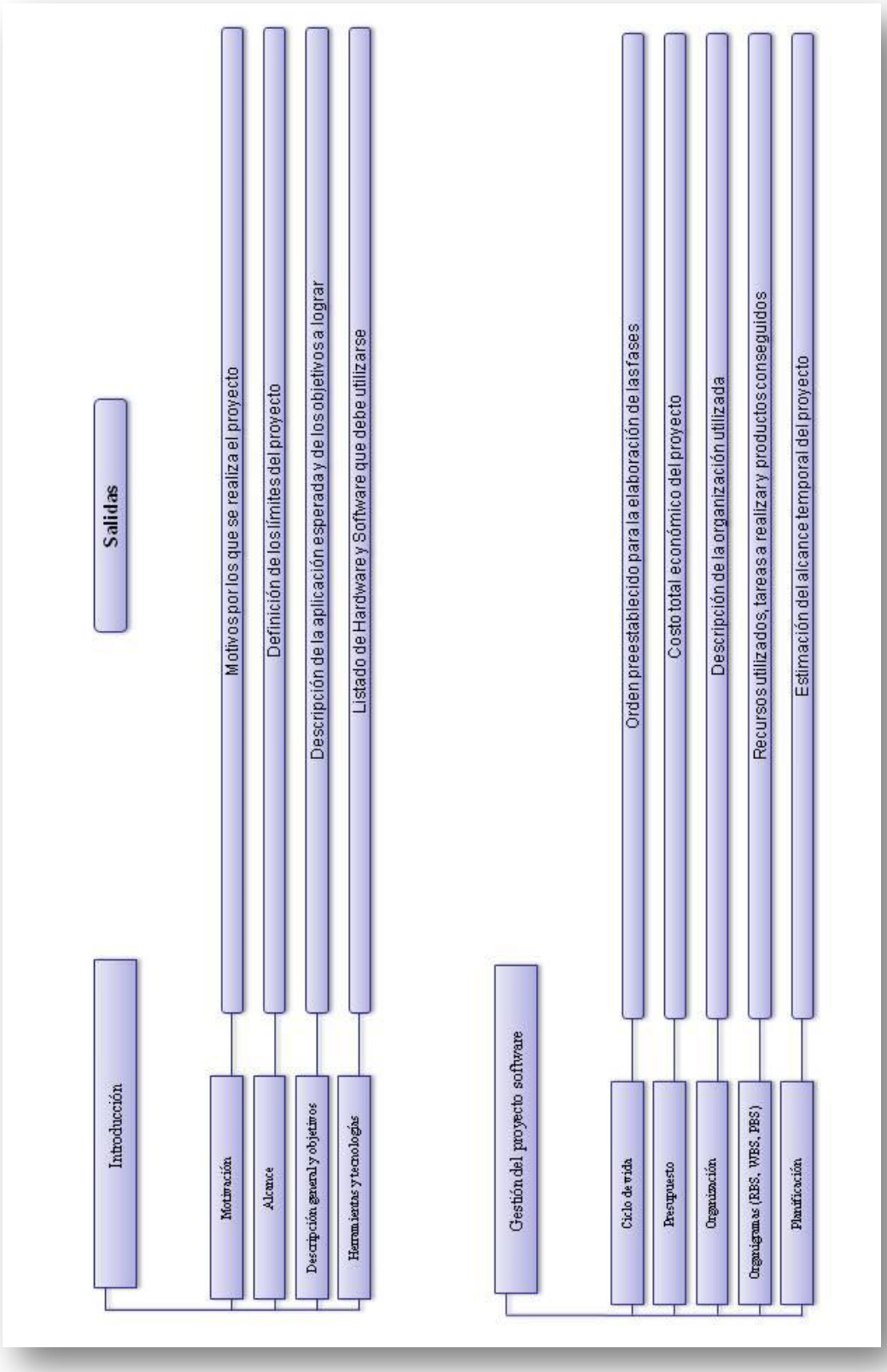


Ilustración 5: PBS 1

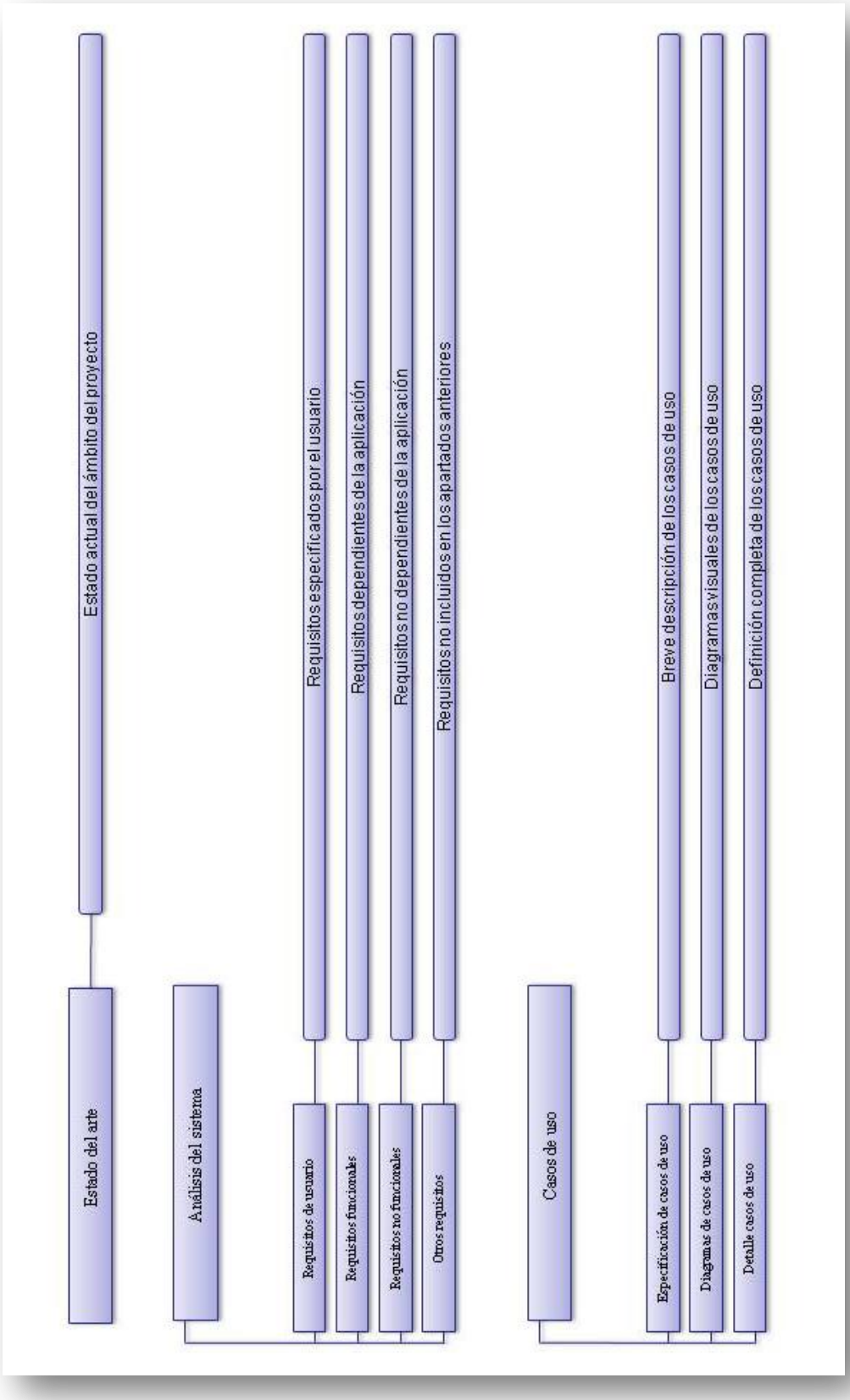


Ilustración 6: PBS 2

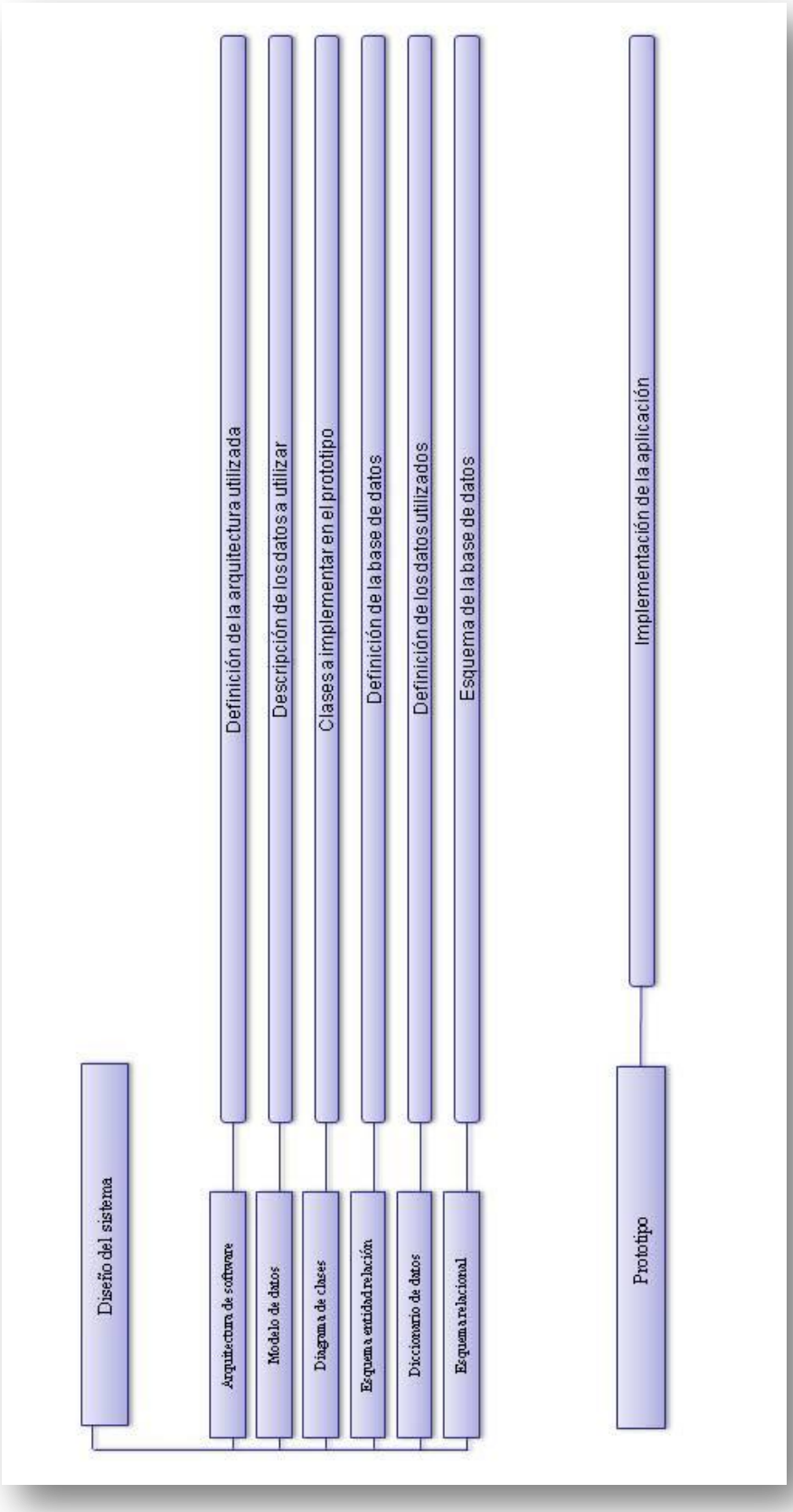


Ilustración 7: PBS 3

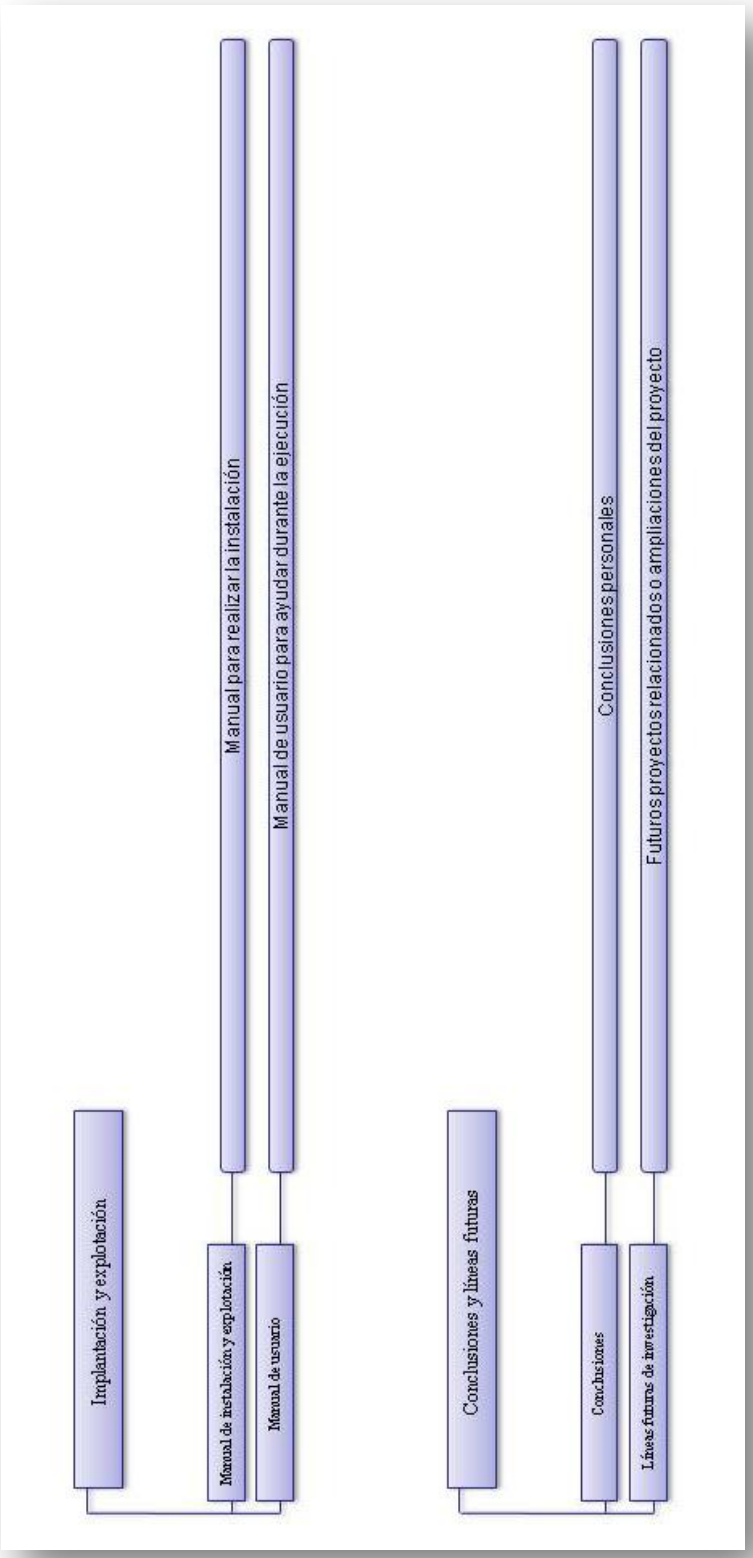
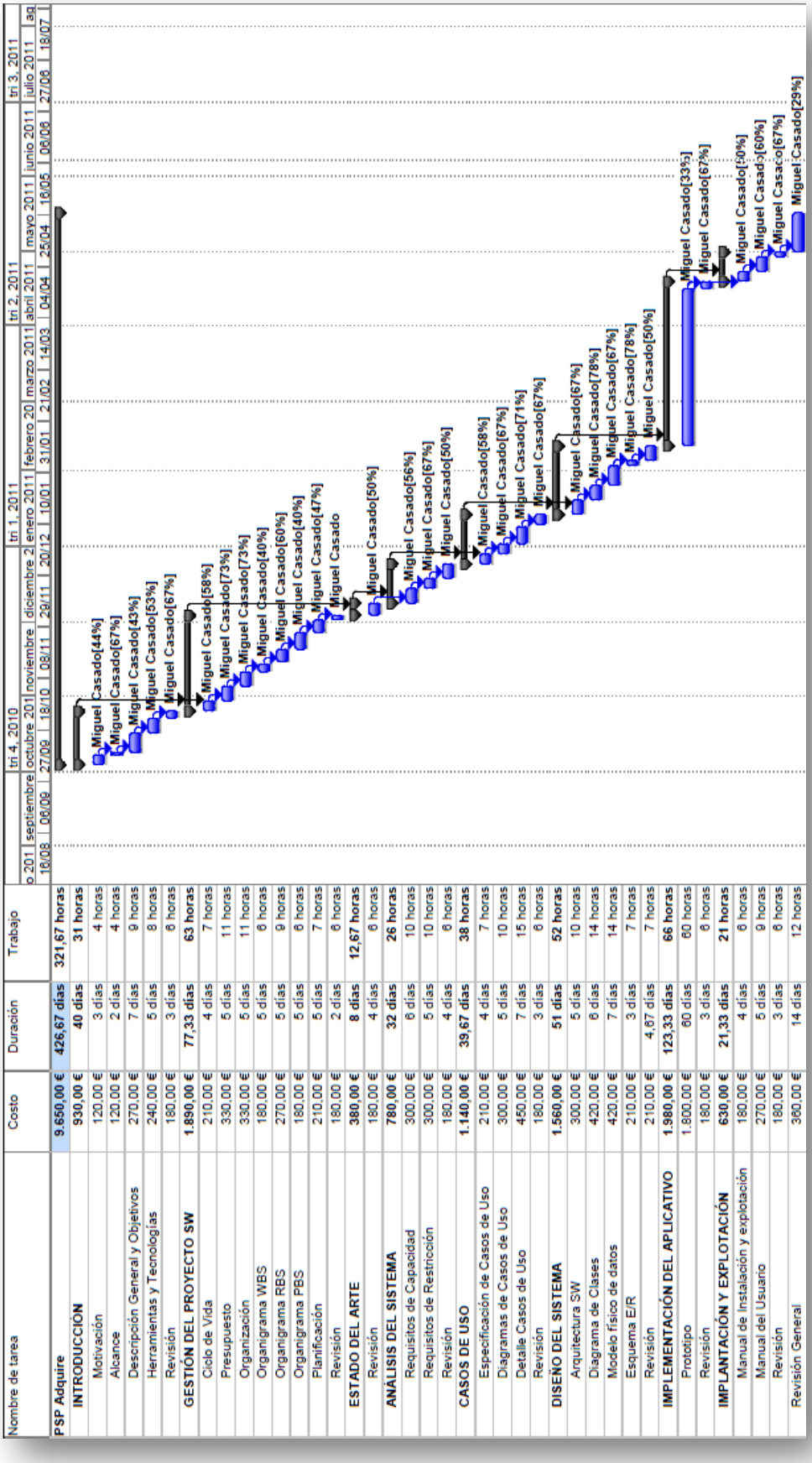


Ilustración 8: PBS 4

Planificación:

La planificación del proyecto se ha realizado utilizando Microsoft Project, y realizando una estimación optimista sobre el tiempo necesario en cada fase. Dicha estimación se ha realizando extrapolando los tiempos utilizados en dichas fases durante proyectos anteriores.

El diagrama de Gantt queda como sigue, realizando una estimación de fin para el 16/05/2011.



Nombre de tarea	Costo	Duración	Trabajo
PSP Adquire	9.650,00 €	426,67 días	321,67 horas
INTRODUCCIÓN	930,00 €	40 días	31 horas
Motivación	120,00 €	3 días	4 horas
Alcance	120,00 €	2 días	4 horas
Descripción General y Objetivos	270,00 €	7 días	9 horas
Herramientas y Tecnologías	240,00 €	5 días	8 horas
Revisión	180,00 €	3 días	6 horas
GESTIÓN DEL PROYECTO SW	1.890,00 €	77,33 días	63 horas
Ciclo de Vida	210,00 €	4 días	7 horas
Presupuesto	330,00 €	5 días	11 horas
Organización	330,00 €	5 días	11 horas
Organigrama WBS	180,00 €	5 días	6 horas
Organigrama RBS	270,00 €	5 días	9 horas
Organigrama PBS	180,00 €	5 días	6 horas
Planificación	210,00 €	5 días	7 horas
Revisión	180,00 €	2 días	6 horas
ESTADO DEL ARTE	380,00 €	8 días	12,67 horas
Revisión	180,00 €	4 días	6 horas
ANÁLISIS DEL SISTEMA	780,00 €	32 días	26 horas
Requisitos de Capacidad	300,00 €	6 días	10 horas
Requisitos de Restricción	300,00 €	5 días	10 horas
Revisión	180,00 €	4 días	6 horas
CASOS DE USO	1.140,00 €	39,67 días	38 horas
Especificación de Casos de Uso	210,00 €	4 días	7 horas
Diagramas de Casos de Uso	300,00 €	5 días	10 horas
Detalle Casos de Uso	450,00 €	7 días	15 horas
Revisión	180,00 €	3 días	6 horas
DISEÑO DEL SISTEMA	1.560,00 €	51 días	52 horas
Arquitectura SW	300,00 €	5 días	10 horas
Diagrama de Clases	420,00 €	6 días	14 horas
Modelo físico de datos	420,00 €	7 días	14 horas
Esquema E/R	210,00 €	3 días	7 horas
Revisión	210,00 €	4,67 días	7 horas
IMPLEMENTACIÓN DEL APLICATIVO	1.980,00 €	123,33 días	66 horas
Prototipo	1.800,00 €	60 días	60 horas
Revisión	180,00 €	3 días	6 horas
IMPLANTACIÓN Y EXPLOTACIÓN	630,00 €	21,33 días	21 horas
Manual de Instalación y explotación	180,00 €	4 días	6 horas
Manual del Usuario	270,00 €	5 días	9 horas
Revisión	180,00 €	3 días	6 horas
Revisión General	360,00 €	14 días	12 horas

Ilustración 9: Diagrama de Gantt

Estado del arte:

Las investigaciones en este tema hasta el momento, se dividen en tres generaciones. En la tabla siguiente se muestran algunas de sus características.

La primera generación hacía uso del PSP como se describe originalmente por su creador: los usuarios creaban e imprimían tablas o formularios que llenaban de forma manual con gran esfuerzo y cansancio por lo trabajoso del llenado de datos.

En la segunda generación se usaban herramientas automatizadas como Leap, PSP Studio o PSP Dashboard. Estas básicamente tienen las mismas prestaciones: cajas de diálogo donde el usuario registra los datos de esfuerzo, tamaño y defectos, aunque también se pueden mostrar análisis si el usuario lo desea. Esta segunda generación, sin dudas minimiza el trabajo del usuario en la recolección de datos así como en su análisis para la toma de decisiones.

Con el desarrollo de Hackystat en Mayo del 2001, el tema evoluciona hacia la 3ra generación con una herramienta que colecciona automáticamente las métricas mediante sensores adjuntos a las herramientas de desarrollo, los datos son enviados por los sensores al servidor y independencia de los análisis pueden ser enviados mensajes de alerta, por ejemplo, a los desarrolladores.

Características:

	1ra Generación	2da Generación	3ra generación
Dificultad en la recopilación de datos	Alta	Media	Nula
Dificultad para el análisis	Alta	Baja	Nula
Cambios en las métricas	Simple	Software	Dependiente de la herramienta

Tabla 10: Generaciones de PSP

El proyecto que nos ocupa se puede enmarcar dentro de la 2ª generación de software, ya que supone una mejora desde la 1ª Generación y evita los problemas generados por la 3ª generación debidos a su complejidad.

El **Model-View-Controller** (Modelo-Vista-Controlador, en adelante MVC) fue introducido inicialmente en la comunidad de desarrolladores de Smalltalk-80. Según uno de los Sistemas de Patrones de Arquitectura más extendido en el mundo: *Pattern Oriented Software Architecture*, publicado por Buschmann en 1996, el patrón que se analiza en este trabajo (MVC), se sitúa en la tercera de las cuatro categorías en las cuales clasifica a los patrones:

- Del barro a la estructura (From mud to Structure).
- Sistemas Distribuidos (Distributed Systems).
- Sistemas Interactivos (Interactive Systems).
- Sistemas Adaptables (Adaptable Systems).

De igual forma si utilizamos otro de los sistemas más difundidos: *Pattern of Enterprise Application Architecture*, descrito recientemente por Fowler en el pasado 2003, lo ubica en la tercera de las siete categorías que se mencionan a continuación:

- Patrones de lógica del dominio (Domain Logia Patterns).
- Patrones de Mapeo a Bases de Datos Relacionales (Mapping to Relational Database Patterns).
- Patrones de Presentación Web (Web Presentation Patterns).
- Patrones de Distribución (Distribution Patterns).
- Patrones de Concurrencia Offline (Offline Concurrency Patterns).
- Patrones de Estado de Sesión (Session State Patterns).
- Patrones Base (Base Patterns).

MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- **Modelo (Model)**: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- **Vista (View)**: Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- **Controlador (Controller)**: Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("**service requests**") para el modelo o la vista.

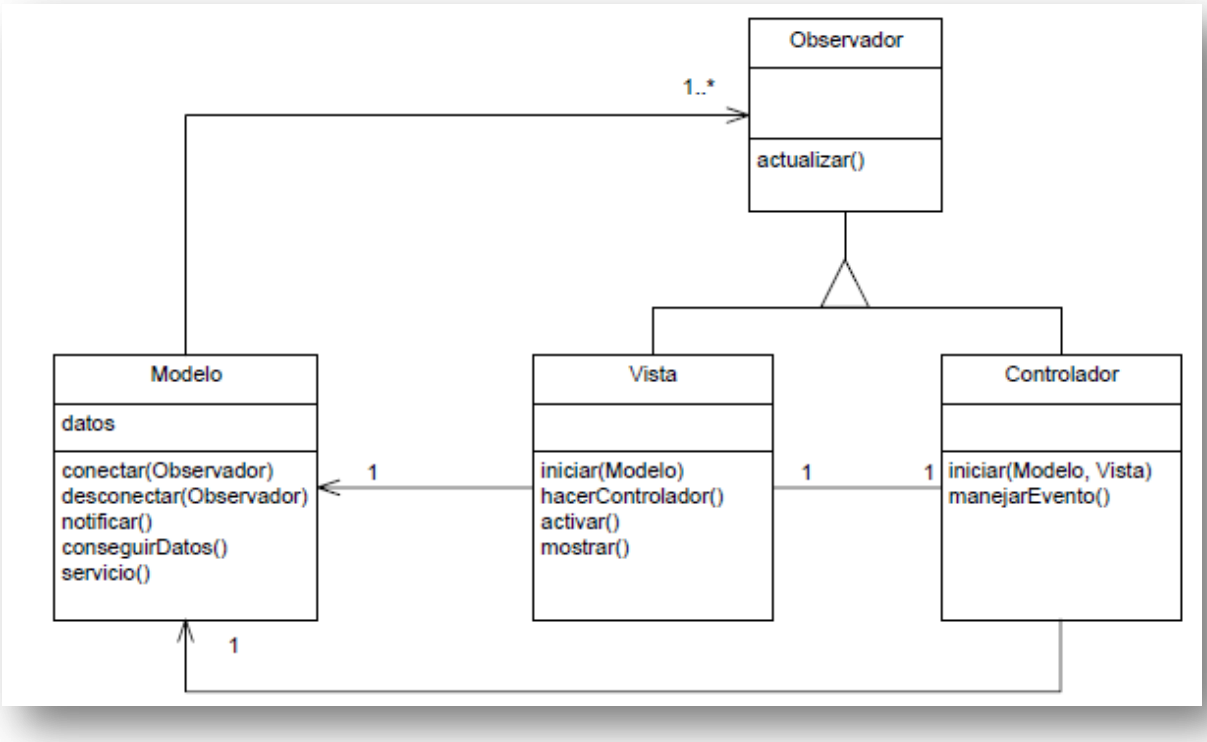


Ilustración 10: Modelo Vista Controlador

Ventajas:

- Múltiples vistas del mismo modelo
- Vistas sincronizadas
- Cambios de vistas y controles en tiempo de ejecución
- Cambio del aspecto externo de las aplicaciones
- Base potencial para construir un Framework

Inconvenientes:

- Se incrementa la complejidad
- Número de actualizaciones potencialmente alto
- Íntima conexión entre la vista y el controlador
- Alto acoplamiento de las vistas y los controladores con respecto al modelo
- Acceso ineficiente a los datos desde la vista
- Cambio inevitable de las vistas y controladores cuando se porte a otras
- Plataformas

ANÁLISIS

Análisis del sistema:

En este punto se procede a la definición de requisitos, cada requisito se define unívocamente a través de la siguiente tabla:

Nombre	El nombre del requisito se compone de la siguiente manera: R[CAP/RES] - XX R: Indicando que es un requisito. CAP: Si el requisito es de capacidad. RES: Si el requisito es de restricción. XX: Número correspondiente al orden del requisito dentro de su categoría.
Descripción	Pequeña descripción del requisito
Actor	Identifica al usuario responsable de la definición del Requisito.
Necesidad	Mide el nivel de importancia del requisito de acuerdo a los siguientes valores: Esencial. Es imprescindible su implementación. Deseable. Debería ser implementado si no conlleva una dificultad o coste excesivo. Opcional. Sería interesante para el usuario su implementación.
Prioridad	Para facilitar una planificación del desarrollo es necesario asignar una prioridad a cada uno de los Requisitos de Usuario. Esta valoración permitirá al desarrollador programar su agenda de acuerdo a dicho valor. Dicha medida será: Alta, Media o Baja.
Disponibilidad	El usuario debe valorar las consecuencias de la perdida de disponibilidad, o violaciones de seguridad, de tal manera que se pueda valorar la criticidad de la función asociada. Dicha medida será: Imprescindible, Necesario o Deseable.
Estabilidad	Permite representar el grado en el que se espera que el requisito no sufra modificaciones que afecten al desarrollo del proyecto. Dicha medida será: Alta, Media o Baja.
Aplicable	Permite marcar aquellos Requisitos de Usuario que no serán aplicados.

Tabla 11: Definición del formato de los requisitos

Requisitos de capacidad:

RCAP-01	
Descripción	El sistema debe funcionar en equipos con sistema operativo Windows XP
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 12: RCAP-01

RCAP-02	
Descripción	El sistema debe poseer una interfaz de usuario simple e intuitivo.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 13: RCAP-02

RCAP-03	
Descripción	La toma de datos debe ser automática, es decir, debe implicar la mínima participación por parte del usuario posible.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 14: RCAP-03

RCAP-04	
Descripción	Identificación del usuario. El sistema debe integrar una validación de usuario para comenzar su uso, así como para garantizar la confidencialidad de los datos.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 15: RCAP-04

RCAP-05	
Descripción	El sistema debe ser multilenguaje tanto en la aplicación, como en los mensajes textuales y la ayuda.
Actor	Usuario
Necesidad	Deseable
Prioridad	Baja
Disponibilidad	Prescindible
Estabilidad	Media
Aplicable	No

Tabla 16: RCAP-05

RCAP-06	
Descripción	El sistema debe generar informes en un fichero externo compatible con herramientas ofimáticas
Actor	Usuario
Necesidad	Deseable
Prioridad	Baja
Disponibilidad	Prescindible
Estabilidad	Baja
Aplicable	No

Tabla 17: RCAP-06

RCAP-07	
Descripción	El sistema debe integrar un subsistema de gestión de Programas
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 18: RCAP-07

RCAP-08	
Descripción	El sistema debe integrar un subsistema de gestión de Proyectos
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 19: RCAP-08

RCAP-09	
Descripción	El sistema debe integrar un subsistema de gestión de Usuarios
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 20: RCAP-09

RCAP-10	
Descripción	El sistema debe integrar un subsistema de gestión de Actividades
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 21: RCAP-10

RCAP-11	
Descripción	El sistema debe integrar un subsistema de gestión de Defectos
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 22: RCAP-11

RCAP-12	
Descripción	El sistema debe integrar un subsistema de gestión de Tiempos
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 23: RCAP-12

RCAP-13	
Descripción	El sistema debe ser capaz de realizar un resumen semanal de actividades gracias a los datos introducidos.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 24: RCAP-13

RCAP-14	
Descripción	El subsistema encargado de la toma de datos debe ejecutarse en segundo plano durante la realización de la tarea.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 25: RCAP-14

Requisitos de restricción:

RRES-01	
Descripción	El sistema debe ser independiente en sí mismo de otras aplicaciones, por tanto se deben añadir las librerías necesarias.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 26: RRES-01

RRES -02	
Descripción	El sistema debe poseer un programa de instalación capaz de insertar librerías externas necesarias para su uso.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 27: RRES-02

RRES -03	
Descripción	La documentación de usuario debe ser multilenguaje.
Actor	Usuario
Necesidad	Opcional
Prioridad	Baja
Disponibilidad	Prescindible
Estabilidad	Alta
Aplicable	No

Tabla 28: RRES-03

RRES -04	
Descripción	Deben realizarse buenas prácticas en la realización del proyecto, para minimizar la tasa de errores de la aplicación
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Baja
Aplicable	Si

Tabla 29: RRES-04

RRES -05	
Descripción	El sistema debe utilizar el menor número de recursos posible
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Baja
Aplicable	Si

Tabla 30: RRES-05

RRES -06	
Descripción	El sistema debe ser seguro y asegurar la confidencialidad de los datos de los usuarios.
Actor	Usuario
Necesidad	Esencial
Prioridad	Alta
Disponibilidad	Imprescindible
Estabilidad	Alta
Aplicable	Si

Tabla 31: RRES-06

Casos de uso:

Actores

Administrador

Es un usuario con privilegios. En el entorno en el que trabajamos se puede considerar que es el jefe de proyecto, puesto que puede realizar labores de seguimiento sobre sus empleados, tomar decisiones sobre los proyectos, así como realizar la asignación de los programas a los usuarios. Tiene todas las funcionalidades propias de un usuario y las suyas propias como administrador.

Usuario

Es el usuario básico del programa. En el entorno en el que trabajamos se puede considerar como usuario a un empleado del jefe de proyecto, un programador.

Diagrama de caso de uso:

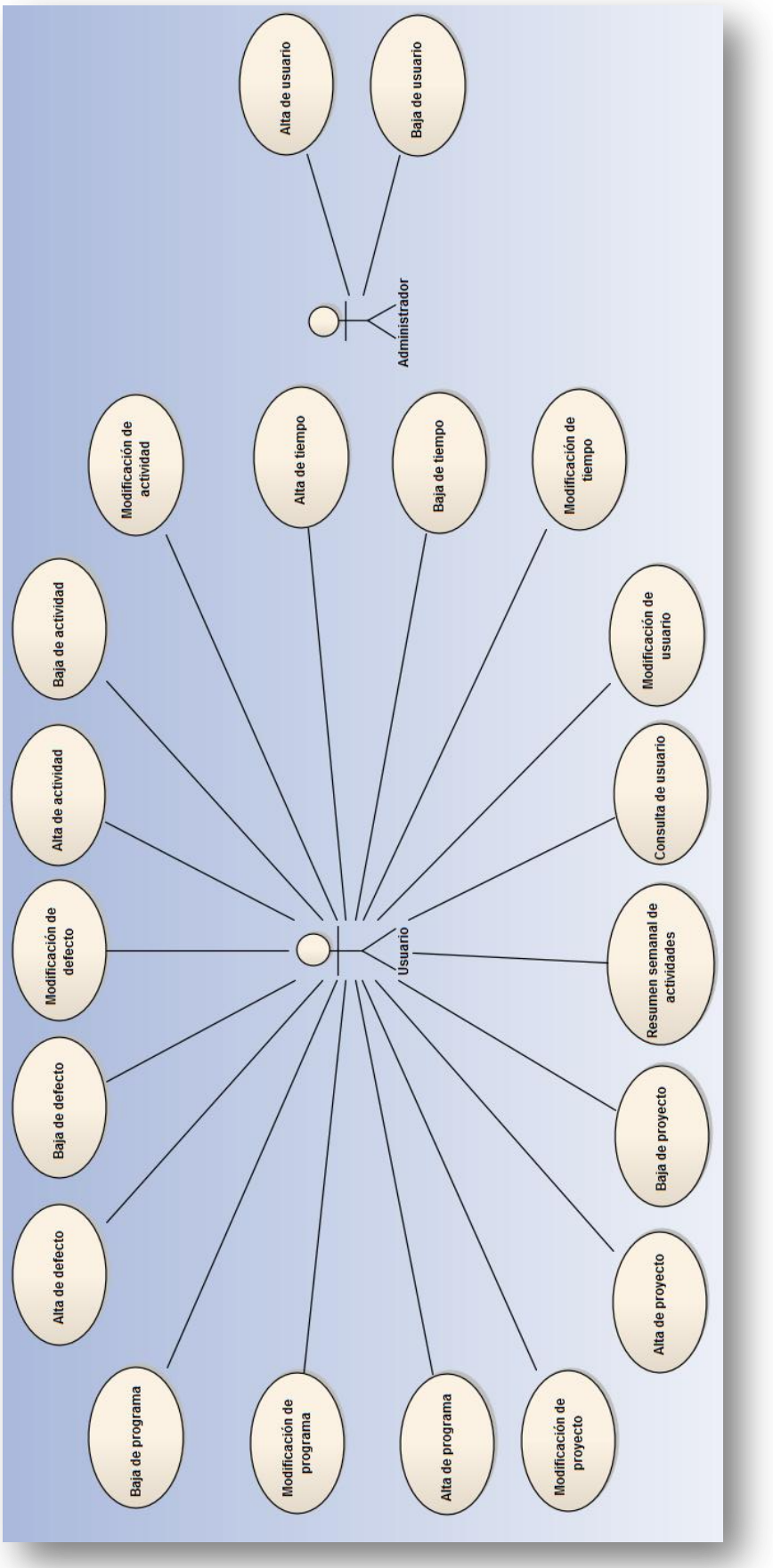


Ilustración 11: Diagrama de Casos de uso

Especificación de los casos de uso:

Para realizar la especificación de cada caso de uso se utilizará la siguiente tabla:
Identificador

Identificador
Nombre
Descripción
Actor
Precondiciones
Postcondiciones
Escenario
Escenario Alternativo

Tabla 32: Especificación de un caso de uso

Donde cada campo significa lo siguiente:

- **Identificador:** Código que identifica de forma unívoca cada caso de uso.
- **Nombre:** Nombre descriptivo del caso de uso.
- **Descripción:** Explicación clara y concisa del caso de uso que se está especificando.
- **Actor:** tipo de usuario de la aplicación.
- **Precondiciones:** Condiciones que se deben cumplir previamente para poder realizar una determinada operación.
- **Postcondiciones:** Estado que presenta el sistema tras la ejecución de una determinada operación.
- **Escenario:** Flujo de ejecución del caso de uso.
- **Escenarios Alternativos:** Flujo de ejecución afectado por condiciones excepcionales.

CAS-01	
Nombre	Alta de actividad.
Descripción	Permite dar de alta una nueva actividad en el sistema. Dicha actividad no tiene que existir previamente en la base de datos, y el sistema debe comprobar que no es una de las actividades definidas por el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor introduce los datos de la actividad. • El sistema confirma la validez de los datos. • El sistema genera un identificador único para la actividad. • El sistema introduce la actividad en la base de datos. • El sistema asocia la actividad nueva al usuario.
Escenario Alternativo	• El actor introduce los datos de la actividad. • El sistema confirma la validez de los datos. • Si los datos son incorrectos el sistema muestra un mensaje informativo y se muestra de nuevo la pantalla de introducción de los datos.

Tabla 33: CAS-01

CAS-02	
Nombre	Baja de actividad.
Descripción	Permite dar de baja una actividad en el sistema. Dicha actividad tiene que existir previamente en la base de datos, y el sistema debe comprobar que no es una de las actividades definidas por el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor elige una o varias actividades de la lista de actividades. • El actor confirma el borrado de las actividades seleccionadas. • El sistema borra las actividades de la base de datos.
Escenario Alternativo	• El actor elige una o varias actividades de la lista de actividades. • El actor cancela el borrado de las actividades seleccionadas. • El sistema vuelve a mostrar la pantalla principal.

Tabla 34: CAS-02

CAS-03	
Nombre	Modificación de actividad.
Descripción	Permite modificar una actividad en el sistema. Dicha actividad tiene que existir previamente en la base de datos, y el sistema debe comprobar que no es una de las actividades definidas por el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona la actividad a modificar. • El sistema recupera los datos de la actividad y los muestra al actor. • El actor modifica los datos y acepta las modificaciones. • El sistema actualiza la actividad en la base de datos.
Escenario Alternativo	• El actor selecciona la actividad a modificar. • El sistema recupera los datos de la actividad y los muestra al actor. • El actor modifica los datos y cancela las modificaciones. • El sistema vuelve a mostrar la pantalla principal.

Tabla 35: CAS-03

CAS-04	
Nombre	Alta de defecto.
Descripción	Permite dar de alta un nuevo defecto en el sistema. Dicho defecto está asignado a un programa, y por tanto a un usuario (el que está realizando dicho programa).
Actor	Usuario
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor introduce los datos del defecto. • El sistema confirma la validez de los datos. • El sistema introduce el defecto en la base de datos. • El sistema asocia el defecto al programa en el que se ha introducido.
Escenario Alternativo	• El actor introduce los datos del defecto. • El sistema confirma la validez de los datos. • Si los datos son incorrectos el sistema muestra un mensaje informativo y se muestra de nuevo la pantalla de introducción de los datos.

Tabla 36: CAS-04

CAS-05	
Nombre	Baja de defecto.
Descripción	Permite dar de baja defecto en el sistema. Dicho defecto está asignado a un programa, y por tanto a un usuario (el que está realizando dicho programa).
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor elige uno o varios defectos de la lista de defectos. • El actor confirma el borrado de los defectos seleccionados. • El sistema borra los defectos de la base de datos.
Escenario Alternativo	• El actor elige uno o varios defectos de la lista de defectos. • El actor cancela el borrado de los defectos seleccionados. • El sistema vuelve a mostrar la pantalla principal.

Tabla 37: CAS-05

CAS-06	
Nombre	Modificación de defecto.
Descripción	Permite modificar un defecto en el sistema. Dicho defecto está asignado a un programa, y por tanto a un usuario (el que está realizando dicho programa).
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona el defecto a modificar. • El sistema recupera los datos del defecto y los muestra al actor. • El actor modifica los datos y acepta las modificaciones. • El sistema actualiza el defecto en la base de datos.
Escenario Alternativo	• El actor selecciona el defecto a modificar. • El sistema recupera los datos del defecto y los muestra al actor. • El actor modifica los datos y cancela las modificaciones. • El sistema vuelve a mostrar la pantalla principal.

Tabla 38: CAS-06

CAS-07	
Nombre	Alta de programa
Descripción	Permite dar de alta un nuevo programa en el sistema. Dicho programa está asociado a un proyecto ya existente. El programa queda asignado al usuario que lo da de alta en el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor introduce los datos del programa. • El sistema confirma la validez de los datos. • El sistema introduce el programa en la base de datos. • El programa se asocia al usuario que lo da de alta en el sistema.
Escenario Alternativo	• El actor introduce los datos del programa. • El sistema confirma la validez de los datos. • Si los datos son incorrectos el sistema muestra un mensaje informativo y se muestra de nuevo la pantalla de introducción de los datos.

Tabla 39: CAS-07

CAS-08	
Nombre	Baja de programa
Descripción	Permite dar de baja un programa en el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor elige uno o varios programas de la lista de programas. • El actor confirma el borrado de los programas seleccionados. • El sistema borra los programas de la base de datos.
Escenario Alternativo	• El actor elige uno o varios programas de la lista de programas. • El actor cancela el borrado de los programas seleccionados. • El sistema vuelve a mostrar la pantalla principal.

Tabla 40: CAS-08

CAS-09	
Nombre	Modificación de programa.
Descripción	Permite modificar los datos de un programa en el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona el programa a modificar. • El sistema recupera los datos del programa y los muestra al actor. • El actor modifica los datos y acepta las modificaciones. • El sistema actualiza el programa en la base de datos.
Escenario Alternativo	• El actor selecciona el programa a modificar. • El sistema recupera los datos del programa y los muestra al actor. • El actor modifica los datos y cancela las modificaciones. • El sistema vuelve a mostrar la pantalla principal.

Tabla 41: CAS-09

CAS-10	
Nombre	Alta de proyecto.
Descripción	Permite dar de alta un nuevo proyecto en el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor introduce los datos del proyecto. • El sistema confirma la validez de los datos. • El sistema introduce el proyecto en la base de datos.
Escenario Alternativo	• El actor introduce los datos del proyecto. • El sistema confirma la validez de los datos. • Si los datos son incorrectos el sistema muestra un mensaje informativo y se muestra de nuevo la pantalla de introducción de los datos.

Tabla 42: CAS-10

CAS-11	
Nombre	Baja de proyecto.
Descripción	Permite dar de baja un proyecto del sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona uno o varios proyectos de la lista de proyectos. • El actor confirma el borrado de los proyectos seleccionados. • El sistema borra los proyectos de la base de datos.
Escenario Alternativo	• El actor selecciona uno o varios proyectos de la lista de proyectos. • El actor cancela el borrado de los proyectos seleccionados. • El sistema vuelve a mostrar la pantalla de baja de proyecto.

Tabla 43: CAS-11

CAS-12	
Nombre	Modificación de proyecto.
Descripción	Permite modificar los datos de un proyecto del sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona el proyecto a modificar. • El sistema recupera los datos del proyecto y los muestra al actor. • El actor modifica los datos y acepta las modificaciones. • El sistema actualiza el proyecto en la base de datos
Escenario Alternativo	• El actor selecciona el proyecto a modificar. • El sistema recupera los datos del proyecto y los muestra al actor. • El actor modifica los datos y cancela las modificaciones. • El sistema vuelve a mostrar la pantalla principal.

Tabla 44: CAS-12

CAS-13	
Nombre	Alta de tiempo.
Descripción	Permite dar de alta un nuevo tiempo en el sistema. Una vez que se realiza la inserción se debe comprobar el tipo de actividad que se ha realizado, si es una de las actividades mínimas del desarrollo de software se debe comprobar que está asociada a un programa que está realizando el usuario.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor utiliza la toma automática de tiempos. • El actor para la toma de tiempos automáticos. • El sistema confirma la validez de los datos • El sistema genera un identificador único para el tiempo • El sistema introduce el tiempo en la base de datos.
Escenario Alternativo	• El actor introduce los datos del tiempo • El sistema confirma la validez de los datos • El sistema genera un identificador único para el tiempo • El sistema introduce el tiempo en la base de datos.
Escenario Alternativo	• El actor introduce los datos del tiempo • El sistema confirma la validez de los datos • Si los datos son incorrectos el sistema muestra un mensaje informativo y se muestra de nuevo la pantalla de introducción de los datos.

Tabla 45: CAS-13

CAS-14	
Nombre	Baja de tiempo.
Descripción	Permite dar de baja un tiempo del sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona uno o varios tiempos de la lista de tiempos. • El actor confirma el borrado de los tiempos seleccionados. • El sistema borra los tiempos de la base de datos.
Escenario Alternativo	• El actor selecciona uno o varios tiempos de la lista de tiempos. • El actor cancela el borrado de los tiempos seleccionados. • El sistema vuelve a mostrar la pantalla principal.

Tabla 46: CAS-14

CAS-15	
Nombre	Modificación de tiempo.
Descripción	Permite dar de baja un tiempo del sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona el tiempo a modificar. • El sistema recupera los datos del tiempo y los muestra al actor. • El actor modifica los datos y acepta las modificaciones. • El sistema actualiza el tiempo en la base de datos.
Escenario Alternativo	• El actor selecciona el tiempo a modificar. • El sistema recupera los datos del tiempo y los muestra al actor. • El actor modifica los datos y cancela las modificaciones. • El sistema vuelve a mostrar la pantalla principal.

Tabla 47: CAS-15

CAS-16	
Nombre	Consulta de usuario
Descripción	Permite consultar un usuario del sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona un usuario de la lista de usuarios. • El sistema recupera los datos del usuario y los muestra al actor.
Escenario Alternativo	

Tabla 48: CAS-16

CAS-17	
Nombre	Modificación de usuario.
Descripción	Permite modificar los datos del usuario validado en el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor modifica los datos y acepta las modificaciones. • El sistema actualiza el usuario en la base de datos.
Escenario Alternativo	• El actor modifica los datos y cancela las modificaciones. • El sistema vuelve a mostrar la pantalla de modificación de usuarios.

Tabla 49: CAS-17

CAS-18	
Nombre	Cambiar contraseña.
Descripción	Permite modificar la contraseña del usuario que está validado en el sistema.
Actor	Usuario.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor introduce la contraseña antigua y la contraseña nueva. • El sistema comprueba la contraseña antigua y muestra un mensaje de confirmación. • El actor confirma la modificación de la contraseña. • El sistema actualiza la contraseña del usuario.
Escenario Alternativo	• El actor introduce la contraseña antigua y la contraseña nueva. • El sistema comprueba la contraseña antigua y muestra un mensaje de error. • El sistema vuelve a mostrar la pantalla de modificación de la contraseña.
Escenario Alternativo	• El actor introduce la contraseña antigua y la contraseña nueva. • El sistema comprueba la contraseña antigua y muestra un mensaje de confirmación. • El actor cancela la modificación de la contraseña. • El sistema vuelve a mostrar la pantalla de modificación de la contraseña.

Tabla 50: CAS-18

CAS-19	
Nombre	Alta de usuario.
Descripción	Permite dar de alta un nuevo usuario en el sistema.
Actor	Administrador.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor introduce los datos del usuario • El sistema confirma la validez de los datos • El sistema genera un identificador único para el usuario, así como una contraseña. • El sistema introduce el usuario en la base de datos. • El sistema informa al actor de los datos generados para el usuario, mostrándole su identificador y contraseña.
Escenario Alternativo	• El actor introduce los datos del usuario • El sistema confirma la validez de los datos • Si los datos son incorrectos el sistema muestra un mensaje informativo y se muestra de nuevo la pantalla de introducción de los datos.

Tabla 51: CAS-19

CAS-20	
Nombre	Baja de usuario.
Descripción	Permite dar de alta un nuevo usuario en el sistema.
Actor	Administrador.
Precondiciones	El actor debe estar validado en el sistema.
Postcondiciones	Ninguna.
Escenario	• El actor selecciona uno o varios usuarios de la lista de usuarios. • El actor confirma el borrado de los usuarios. • El sistema borra los usuarios de la base de datos.
Escenario Alternativo	• El actor selecciona uno o varios usuarios de la lista de usuarios. • El actor cancela el borrado de los usuarios. • El sistema vuelve a mostrar la pantalla principal.

Tabla 52: CAS-20

DISEÑO

Diseño del sistema:

Arquitectura del software:

La arquitectura en 3 capas o programación en 3 capas consiste en separar un proyecto en:

- Capa de Presentación
- Capa de Negocio
- Capa de Datos**.**

Esto permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

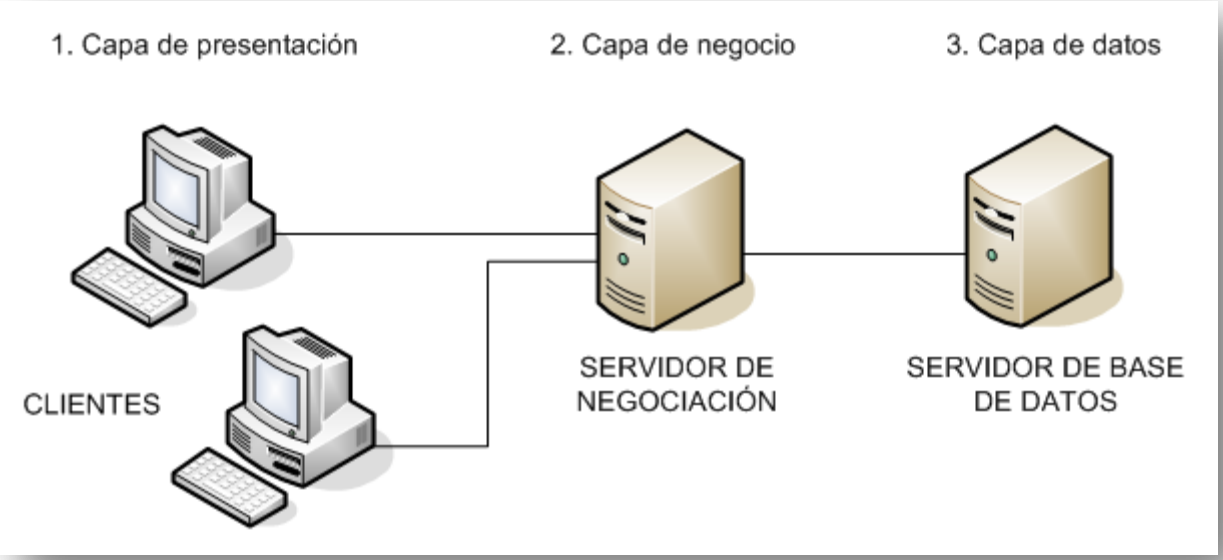


Ilustración 12: Arquitectura de 3 capas

Ventajas de esta Arquitectura:

- El desarrollo se puede llevar a cabo en varios niveles (capas).
- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- En caso de ser necesario algún cambio, sólo se ataca al nivel (capa) requerido sin tener que revisar el resto de los niveles (capas).

- Mantenimiento y soporte más sencillo.
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad)
- Alta escalabilidad.

Capas y niveles

En una arquitectura de tres niveles, los términos “capas” y “niveles” no significan lo mismo ni son similares.

El término “capa” hace referencia a la forma como una solución es segmentada desde el punto de vista lógico:

Presentación - Lógica de Negocio - Datos.

En cambio, el término “nivel” corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física.

En el caso que nos ocupa se utilizará una arquitectura de 3 capas y un único nivel.

- **Capa de Presentación:** Es el interfaz gráfico de la aplicación, se utiliza para que el usuario interaccione con el programa, insertando datos o visualizando los resultados de sus peticiones. El interfaz de usuario debe ser amigable y seguir unos estándares mínimos de calidad, para asegurar el buen funcionamiento de la aplicación. Únicamente puede comunicarse con la capa de negocio, que la facilitará los datos a mostrar, y un API para poder comunicarse y enviar peticiones a la capa de datos.
- **Capa de negocio:** es donde residen los algoritmos que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todos los procesos que deben realizarse, incluidas las validaciones de los datos introducidos por el usuario. Se comunica con las dos capas restantes.
- **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Diagrama de Clases:

Al utilizar un modelo vista controlador quedan separadas tanto la capa visual como la capa lógica de la capa de datos. Como puede observarse en el siguiente diagrama se muestra que no existe relación entre las ventanas (Capa visual) y la capa de datos, que en este caso está representada como adaptadores de datos de .NET. Dichos adaptadores son los que se encargan de proveer los datos de la base de datos a la capa lógica, creando objetos del tipo necesario en el momento oportuno.

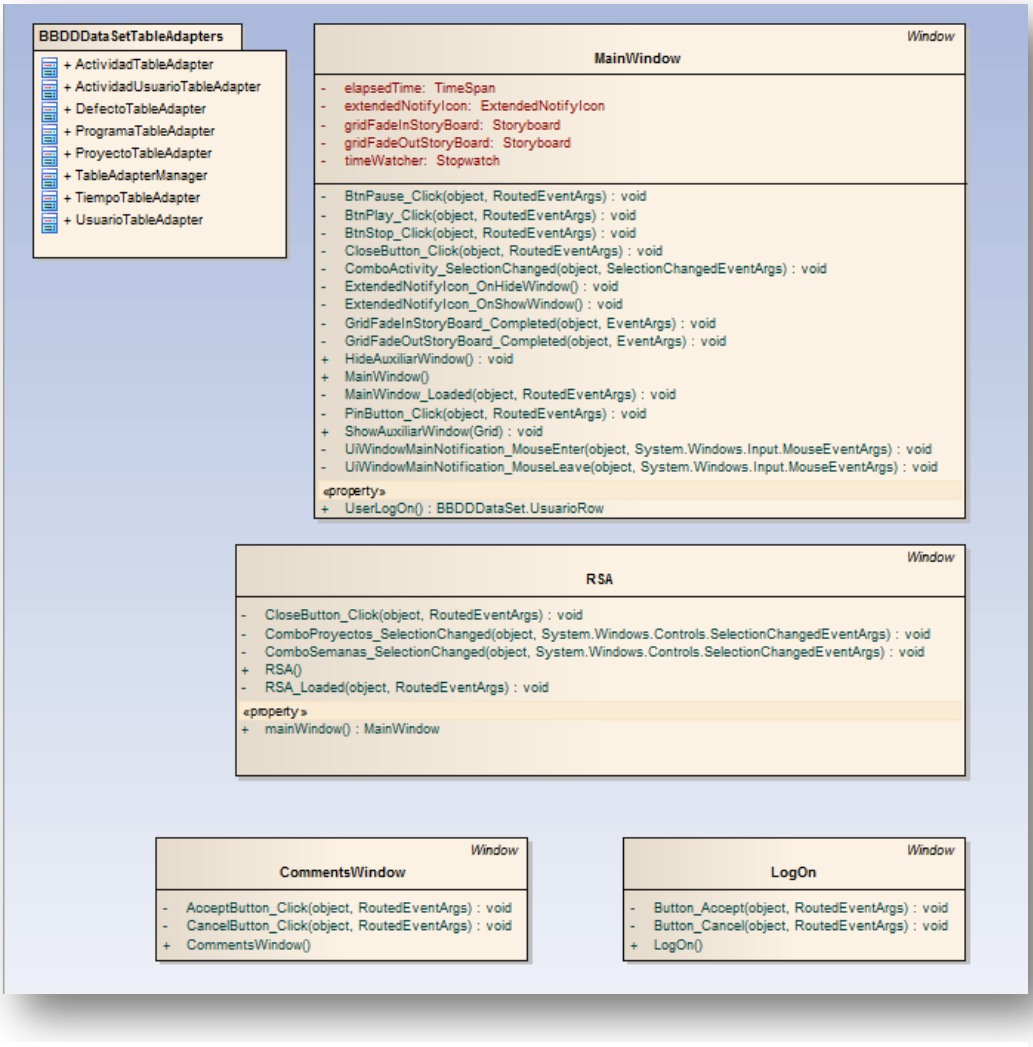


Ilustración 13: Diagrama de clases de ventanas

La clase encargada de controlar el acceso a la base de datos, y que debe ser rellena de datos gracias a los adaptadores de base de datos anteriores es la siguiente. Contiene todos los métodos y datos necesarios para implementar el modelo de datos de la aplicación y su funcionalidad básica. Es generado automáticamente por Visual Studio al añadir una base de datos relacional de Access a nuestro proyecto.

Ilustración 14: Controladora de BBDD

El resto de clases de la aplicación son las que se encargan de interactuar con el usuario (menús) y las que forman parte de la capa lógica (P. Ej.: Alta de Actividad). Debido al gran número de clases se muestra a continuación un ejemplo.

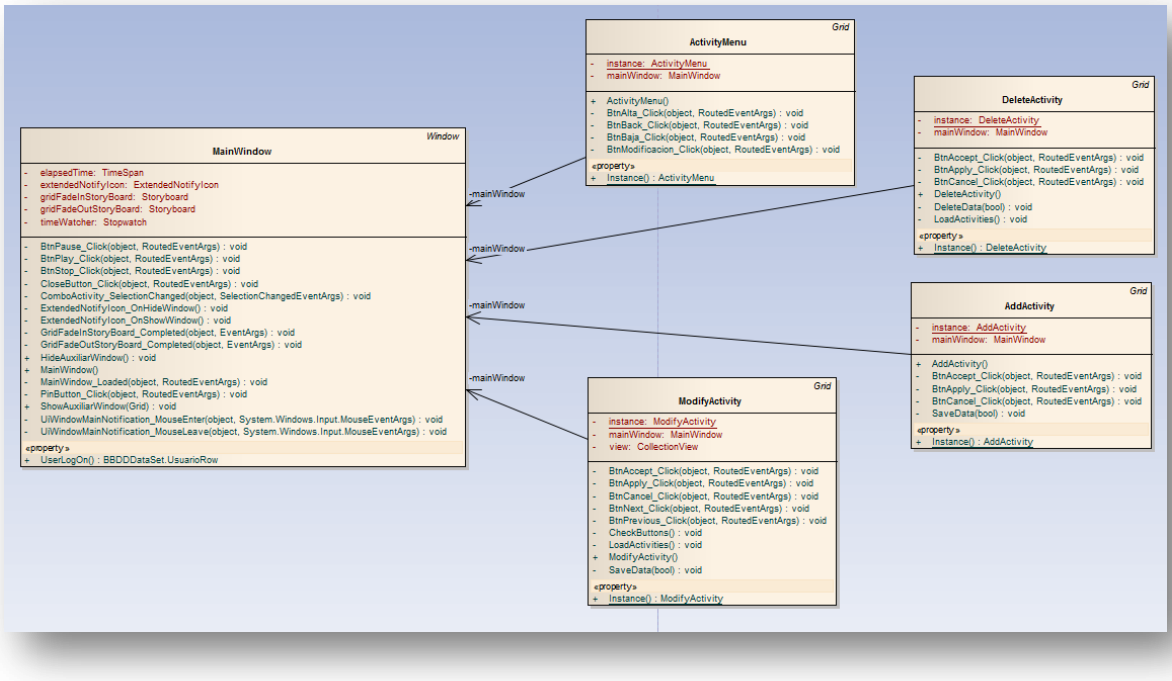


Ilustración 15: Ejemplo de subsistema

Tanto los menús como las clases con lógica de negocio tienen una relación con la ventana principal, puesto que almacenan la instancia de la ventana en la que deben pintarse.

Todas estas clases implementan el patrón de diseño Singleton, por lo que deben saber la instancia concreta de la ventana donde deben pintarse. No tendría lógica que fuese al contrario, es decir, que la ventana padre (MainWindow) tuviera acceso a todas sus ventanas hijas y quedasen almacenadas, puesto que es posible que un usuario no necesite pasar por todas ellas (Véase los subsistemas de la aplicación).

Puesto que no existe la necesidad de crearse una instancia de cada ventana hija se crean dinámicamente gracias al patrón de diseño en el momento que se necesite mostrar dicho menú o capa visual de la lógica de negocio.

La separación entre la capa visual y la capa lógica es realizada por el propio WPF, ya que separa cada clase de las representadas anteriormente (Windows y Grids)

en dos clases parciales, indicando de esta manera que existe una relación entre ellas, pero que están separadas, por lo tanto en la representación del diagrama de clases, cada vez que se represente un objeto visual hay que tener en cuenta que está formado por dos clases parciales.

Modelo físico de datos:

Para la realización del proyecto se va a utilizar el modelo relacional, que define bases de datos relacionales.

Edgar Frank Codd postulo las bases del modelo relacional, su idea fundamental es el uso de relaciones, estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas. Una manera más sencilla de conceptualizar una base de datos relacional a pesar de la teoría formulada, es entender cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia.

La información puede ser recuperada o almacenada mediante consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.
Los sistemas de gestión de base de datos con soporte SQL más utilizados son:

- **DB2**: propiedad de IBM.
- **Oracle**: propiedad de Oracle Corporation, se considera uno de los sistemas de bases de datos más completos, pero su mayor defecto es su elevado precio, dado que es software propietario.
- **SQL Server**: propiedad de Microsoft, no es multiplataforma, ya que sólo está disponible en Sistemas Operativos de Microsoft.
- **Sybase ASE**: propiedad de Sybase, es multiplataforma, existiendo una edición gratuita para Linux, pero con límites de escalabilidad y almacenamiento.
- **MySQL**: se ofrece bajo licencia GNU GPL, es multiplataforma. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no

transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

- **PostgreSQL**: es un motor de base de datos, es servidor de base de datos relacional libre, liberado bajo la licencia BSD. Sus principales características son la alta concurrencia y la amplia variedad de tipos nativos.
- **Firebird**: propiedad de la fundación Mozilla, es un sistema multiplataforma de administración de base de datos relacional de código abierto, basado en la versión 6 de Interbase.
- **Informix**: se trata de una familia de productos de administración de bases de datos relaciones adquirida por IBM.

Sentencias de generación del modelo físico de datos.

Borrado de las tablas existentes

```
DROP TABLE Actividad
;
DROP TABLE ActividadUsuario
;
DROP TABLE Defecto
;
DROP TABLE Programa
;
DROP TABLE Proyecto
;
DROP TABLE Tiempo
;
DROP TABLE Usuario
;
```

Creación de las tablas

```
CREATE TABLE Actividad (
        Id_Actividad Text(10),
        Nombre Text(20),
        Descripcion Text(100),
        Basica YesNo NOT NULL
)
;

CREATE TABLE ActividadUsuario (
        Id_Actividad Text(10),
        Id_Usuario Text(10),
        Id_Programa Text(10)
)
;

CREATE TABLE Defecto (
        Id_Defecto Text(10),
        Id_Programa Text(10),
        Fecha DateTime,
        Numero Long,
        Insertado Text(20),
        Corregido Text(20),
        Tipo Text(15),
        Tiempo Long,
        BCorregido YesNo NOT NULL
)
;
```

```
CREATE TABLE Programa (
        Id_Programa Text(10),
        Id_Usuario Text(10),
        Id_Proyecto Text(10),
        Nombre Text(20),
        Lenguaje Text(20),
        Fecha DateTime,
        Loc_nuevo_cambiado Long
)
;

CREATE TABLE Proyecto (
        Id_Proyecto Text(10),
        Nombre Text(20),
        Descripcion Text(100),
        Fecha_inicio DateTime,
        Fecha_fin DateTime,
        Coste Long
)
;

CREATE TABLE Tiempo (
        Id_Tiempo Text(10),
        Id_Actividad Text(10),
        Id_Usuario Text(10),
        Id_Programa Text(10),
        Fecha DateTime,
        Minutos Long,
        Comentarios Text(200)
)
;
```

```
CREATE TABLE Usuario (
        Id_Usuario Text(10),
        Nombre Text(20),
        Apellidos Text(50),
        DNI Text(9),
        Password Text(15)
)
;
```

Modificación de las claves primarias

```
ALTER TABLE Actividad ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Actividad)
;

ALTER TABLE ActividadUsuario ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Actividad, Id_Usuario, Id_Programa)
;

ALTER TABLE Defecto ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Defecto)
;

ALTER TABLE Programa ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Programa)
;

ALTER TABLE Proyecto ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Proyecto)
;

ALTER TABLE Tiempo ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Tiempo)
;

ALTER TABLE Usuario ADD CONSTRAINT PrimaryKey
        PRIMARY KEY (Id_Usuario)
;
```

Modificación de las claves secundarias.

```
ALTER TABLE ActividadUsuario ADD CONSTRAINT ActividadActividadUsuario
        FOREIGN KEY (Id_Actividad) REFERENCES Actividad (Id_Actividad)
;

ALTER TABLE ActividadUsuario ADD CONSTRAINT ProgramaActividadUsuario
        FOREIGN KEY (Id_Programa) REFERENCES Programa (Id_Programa)
;

ALTER TABLE Defecto ADD CONSTRAINT ProgramaDefecto
        FOREIGN KEY (Id_Programa) REFERENCES Programa (Id_Programa)
;

ALTER TABLE Programa ADD CONSTRAINT ProyectoPrograma
        FOREIGN KEY (Id_Proyecto) REFERENCES Proyecto (Id_Proyecto)
;

ALTER TABLE Tiempo ADD CONSTRAINT ActividadUsuarioTiempo
        FOREIGN KEY (Id_Actividad, Id_Usuario, Id_Programa) REFERENCES
ActividadUsuario (Id_Actividad, Id_Usuario, Id_Programa)
;
```

Esquema Entidad-Relación:

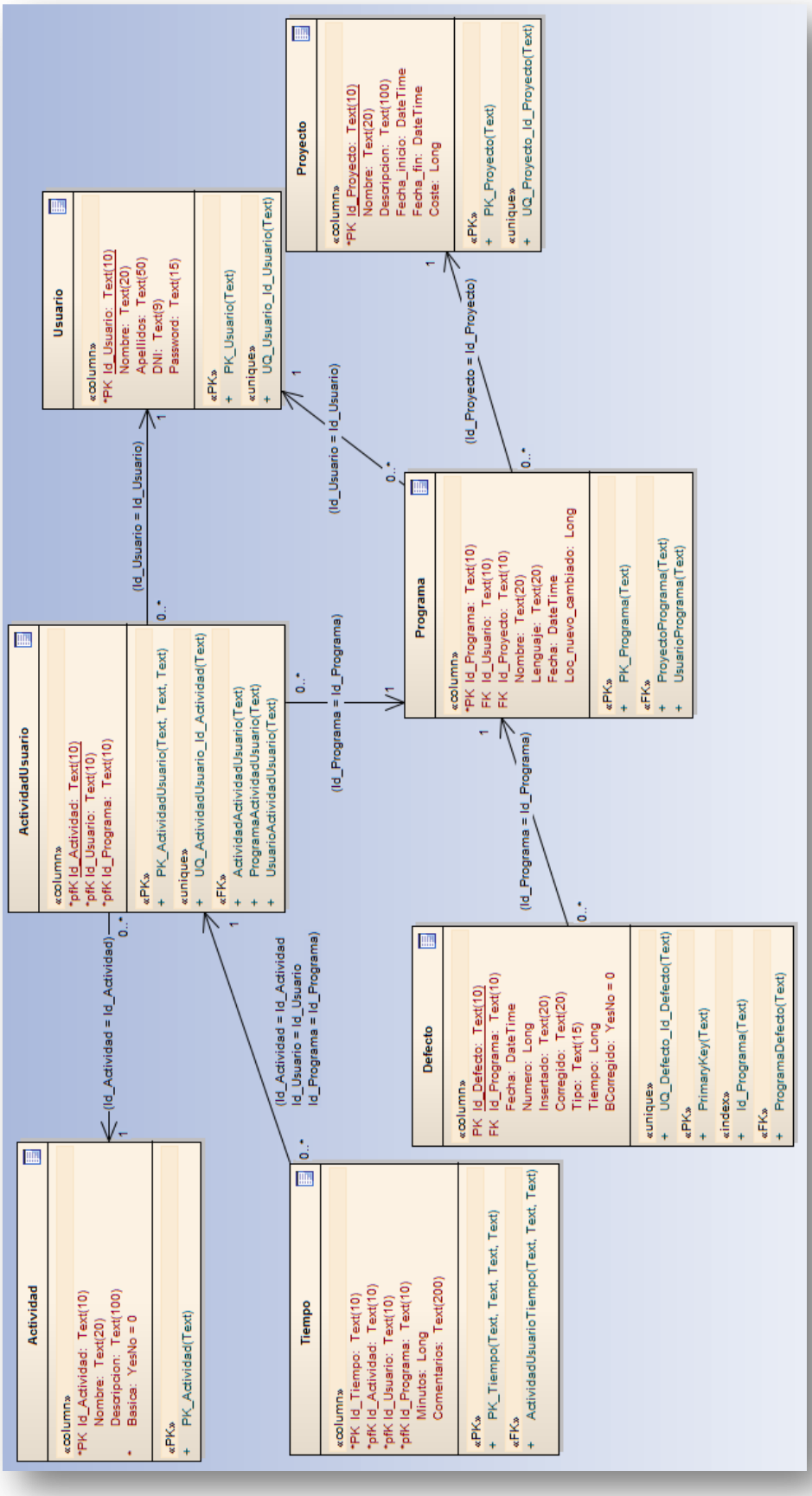


Ilustración 16:Diagrama Entidad / Relación

Implementación:

Una vez expuestos y desarrollados los elementos referentes al análisis y las decisiones tomadas en cuanto al diseño de la aplicación, en este apartado se realizará un estudio en profundidad de las funcionalidades implementadas dando al lector una visión más detallada sobre el funcionamiento interno.

Para conseguir alcanzar este objetivo, en algunos de los siguientes apartados se recurrirá a la inclusión de fragmentos de código fuente para ilustrar las explicaciones, advirtiendo que dicho código no es una copia literal del implementado, ya que incluirlo en su totalidad nos desviaría del carácter informativo que tiene esta sección.

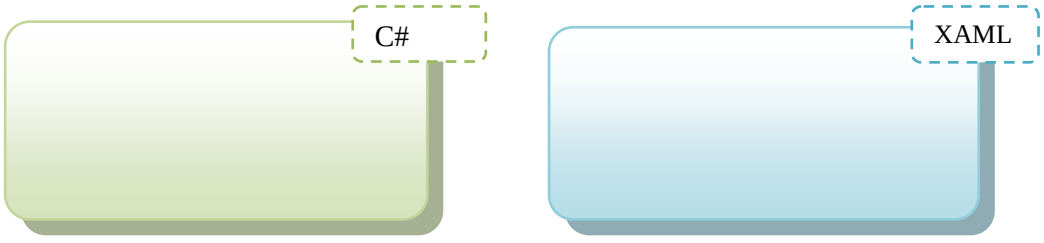
Lenguaje de programación y entorno de desarrollo

Como ya se ha comentado en otros apartados, el lenguaje de programación utilizado para el desarrollo de la aplicación se divide en dos:

- **XAML** (*e**X**tensible **A**pplication **M**arkup **L**anguage)*, utilizado para realizar la capa visual de la aplicación. Este lenguaje es el utilizado por Windows Presentation Foundation y se ha utilizado su versión 4.0.
- **C#** para la capa lógica del sistema. Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.

Se ha utilizado como entorno de desarrollo Visual Studio 2010, puesto que incluye la versión 4.0 de WPF.

Para diferenciar los distintos fragmentos de código mostrados se utilizará la siguiente notación.



Uso de patrones de diseño

Durante el desarrollo de la aplicación se ha utilizado el patrón de diseño singleton, debido a la peculiar manera de representar los menús.

El patrón de diseño Singleton sirve para garantizar que una sólo existe una instancia de una clase concreta y proporcionar un acceso global a dicha instancia.

Puesto que se tiene una ventana principal y los menús son objetos Grid que se van quitando y poniendo según el usuario vaya navegando es necesaria una única instancia de cada objeto visual.

C#

```
public class Singleton
{
    // Variable estática para la instancia, se necesita utilizar una
función lambda ya que el constructor es privado
    private static readonly Lazy<Singleton> instance = new
Lazy<Singleton>(() => new Singleton());

    // Constructor privado para evitar la instanciación directa
    private Singleton()
    {
    }

    // Propiedad para acceder a la instancia
    public static Singleton Instance
    {
        get
        {
            return instance.Value;
        }
    }
}
```

Separación de la capa lógica y la capa visual con la definición de clases parciales.

Durante la definición del diagrama de clases se comentó que la separación de las capas visual y lógica del modelo vista controlador se define gracias a la definición de clases parciales de .NET. El código utilizado para realizar dicha separación es el siguiente. Hay que indicar que dicha separación es realizada automáticamente por Visual Studio, ya que es uno de los paradigmas principales de la tecnología WPF.

Para indicar que son clases parciales se identifican gracias al namespace completo de la clase y del nombre de la clase.

C#

```
namespace Pfc
{
    /// <summary>
    /// Interaction logic for RSA.xaml
    /// </summary>
    public partial class RSA : Window
```

XAML

```
<Window x:Class="Pfc.RSA"
```

Definición de la parte visual

Como se ha comentado para la realización de la parte visual se ha utilizado XAML, más adelante mostraremos un código XAML reducido de la ventana.

La aplicación está dividida en dos partes, una ventana principal, donde se navega por los menús y se puede realizar la toma de tiempos, y una ventana auxiliar, donde se realizan todas las entradas de datos, así como los borrados.

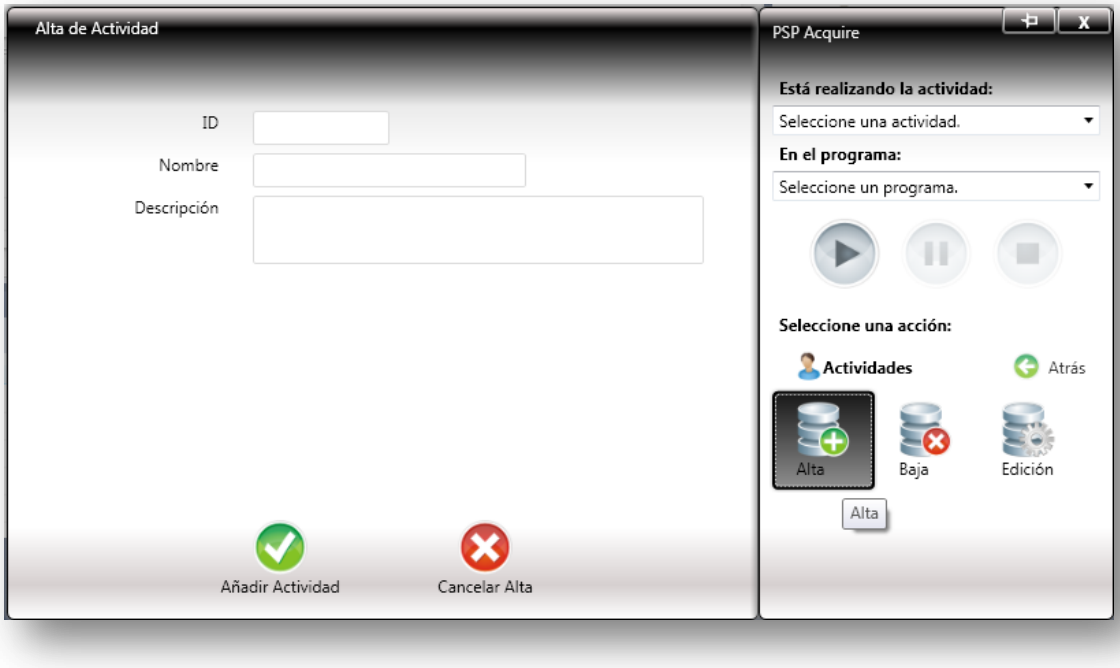


Ilustración 17: Ventanas de la aplicación

VENTANA AUXILIAR

VENTANA

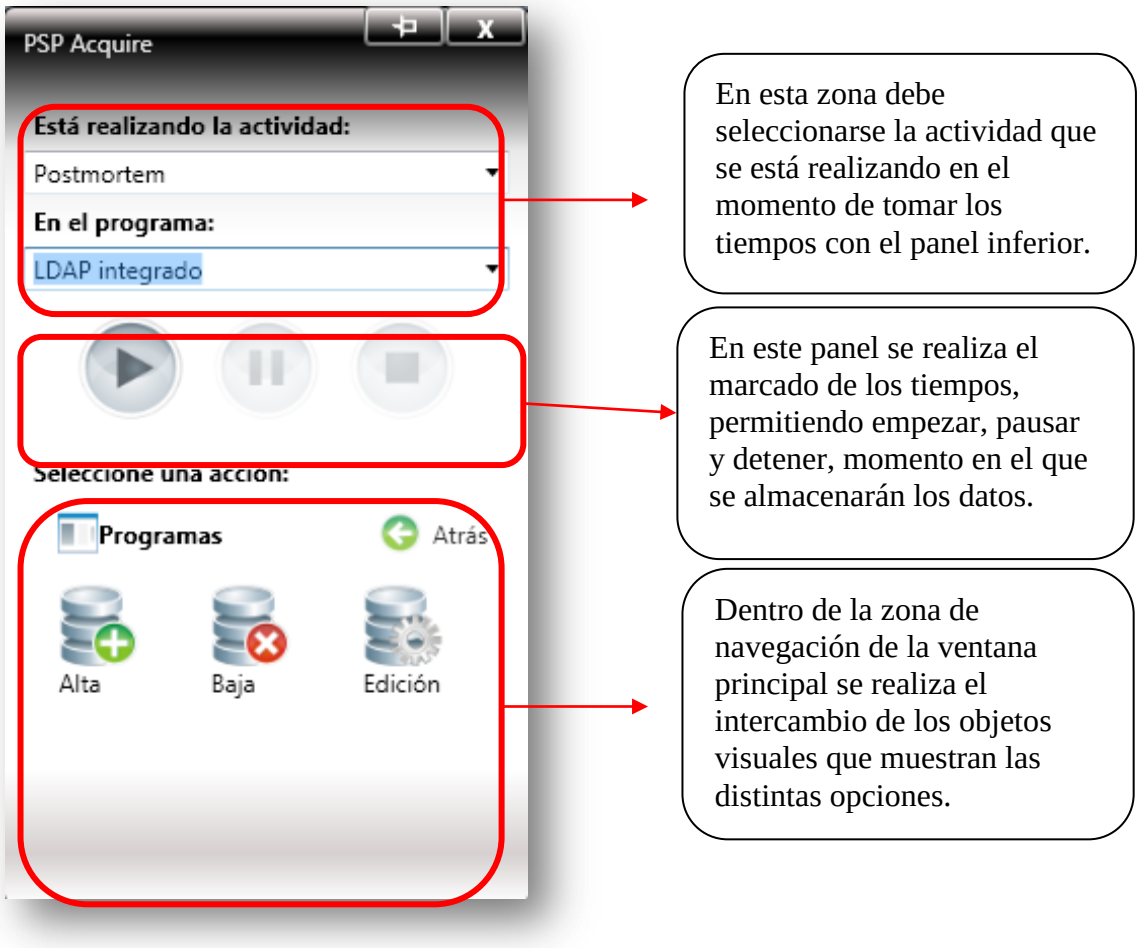


Ilustración 18: Ventana principal

XAML de ejemplo de una opción de menú

XAML

```
<Grid x:Class="Pfc.ActivityMenu"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
    <StackPanel Orientation="Vertical">
      <Grid>
        <StackPanel Orientation="Horizontal" Margin="17,0,0,0">
<Image Source="pack://application:,,,/Images/user.png"
HorizontalAlignment="Left" Height="20"/>
<TextBlock Text="Actividades" Foreground="Black"
VerticalAlignment="Center" FontWeight="Bold"/>
        </StackPanel>
<Button  Click="BtnBack_Click" Name="BtnBack" Width="75"
Style="{StaticResource AcceptButton}" ToolTip="Atrás">
          <StackPanel Orientation="Horizontal" >
<Image Source="pack://application:,,,/Images/back.png" Height="20"/>
            <TextBlock Text="Atrás" Margin="6,0,0,0"/>
          </StackPanel>
       </Button>
      </Grid>
      <WrapPanel>
<Button Name="BtnAlta" Click="BtnAlta_Click" Width="75"
Style="{StaticResource MenuButton}" ToolTip="Alta">
          <StackPanel Orientation="Vertical">
<Image Source="pack://application:,,,/Images/database_add.png"
Height="40"/>
            <TextBlock Text="Alta" />
          </StackPanel>
       </Button>
<Button HorizontalAlignment="Center" Name="BtnBaja"
Click="BtnBaja_Click" Width="75" Style="{StaticResource MenuButton}"
ToolTip="Baja">
          <StackPanel Orientation="Vertical">
<Image Source="pack://application:,,,/Images/database_remove.png"
Height="40"/>
            <TextBlock Text="Baja" />
          </StackPanel>
       </Button>
<Button HorizontalAlignment="Center" Name="BtnModificacion"
Click="BtnModificacion_Click" Width="75" Style="{StaticResource
MenuButton}" ToolTip="Edición">
          <StackPanel Orientation="Vertical">
<Image Source="pack://application:,,,/Images/database_process.png"
Height="40"/>
            <TextBlock Text="Edición" />
          </StackPanel>
       </Button>
      </WrapPanel>
    </StackPanel>
</Grid>
```

Implantación y explotación:

Manual de instalación y explotación:

Para comenzar la instalación se debe hacer doble clic sobre el fichero Setup.exe.

> **Existe un prerrequisito previo en el proceso de instalación, debe tener instalado Microsoft .NET Framework 4.0. En caso de no estar instalado el proceso de instalación mostrará un enlace de descarga.**

Se encuentra disponible para su descarga en el siguiente enlace:
http://go.microsoft.com/fwlink/?LinkId=131000

En esta primera ventana el usuario recibe la bienvenida al proceso de instalación.

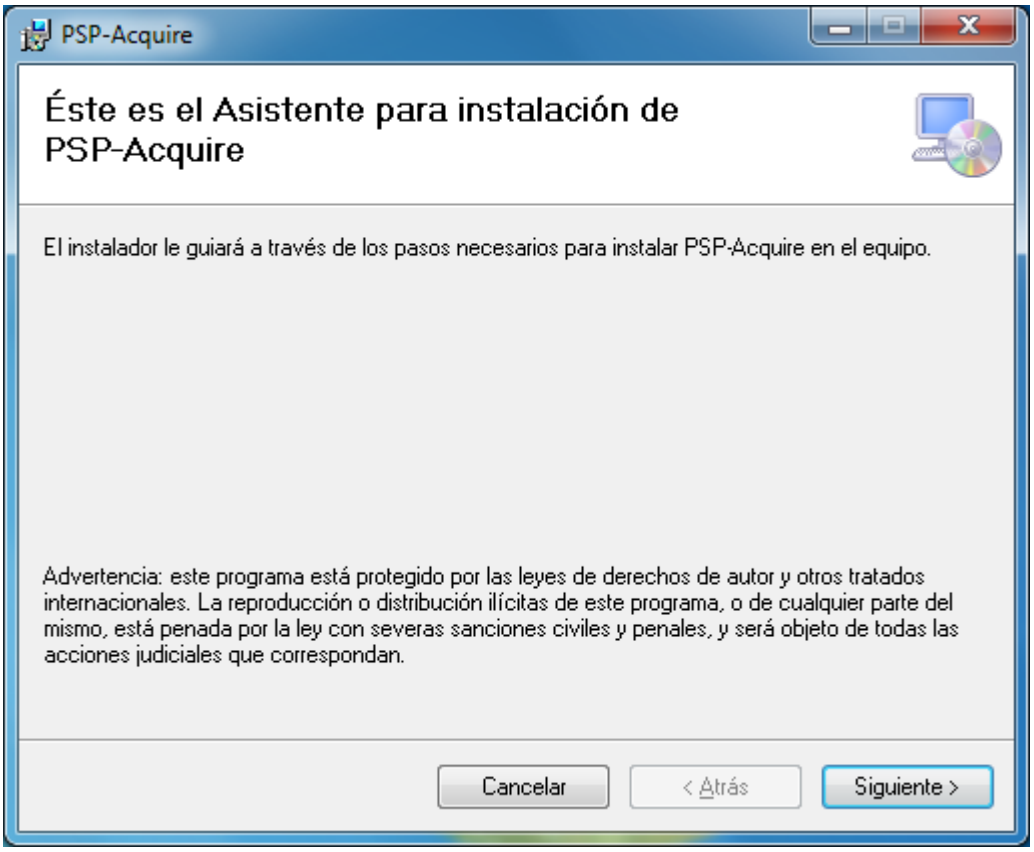


Ilustración 19: Instalación 1

En esta ventana el usuario debe seleccionar la carpeta de instalación.

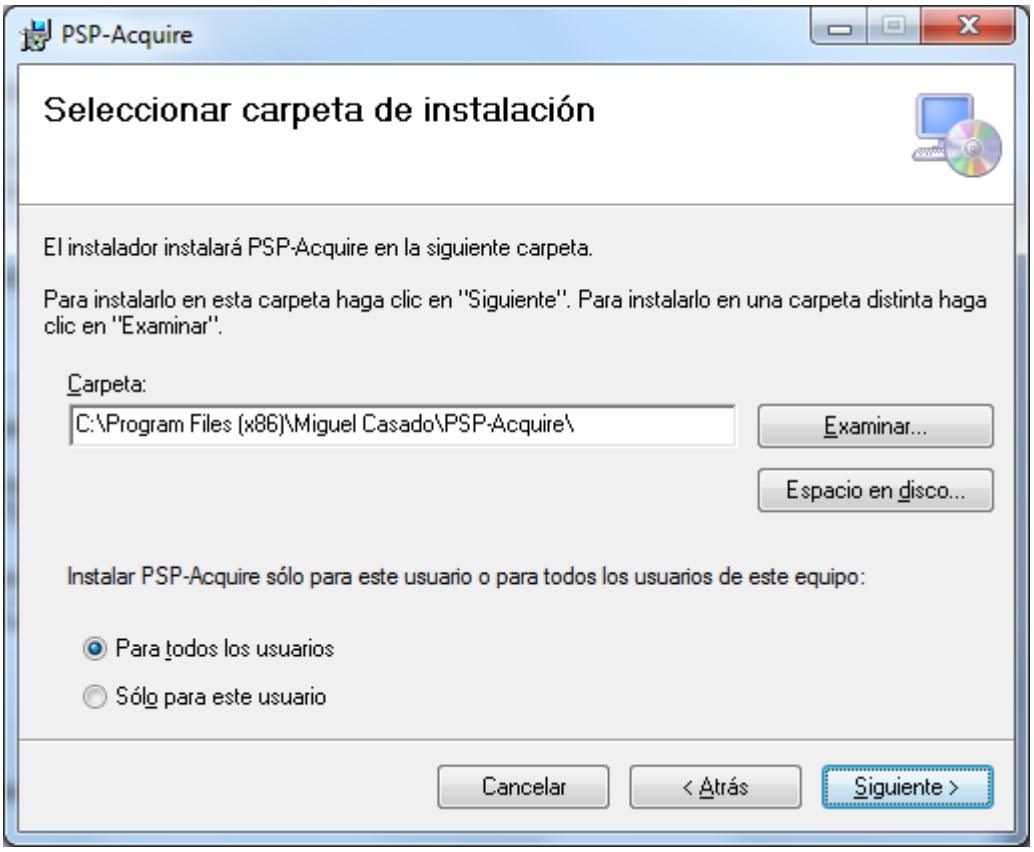


Ilustración 20: Instalación 2

En esta ventana el usuario realiza la confirmación de los datos necesarios para la instalación.

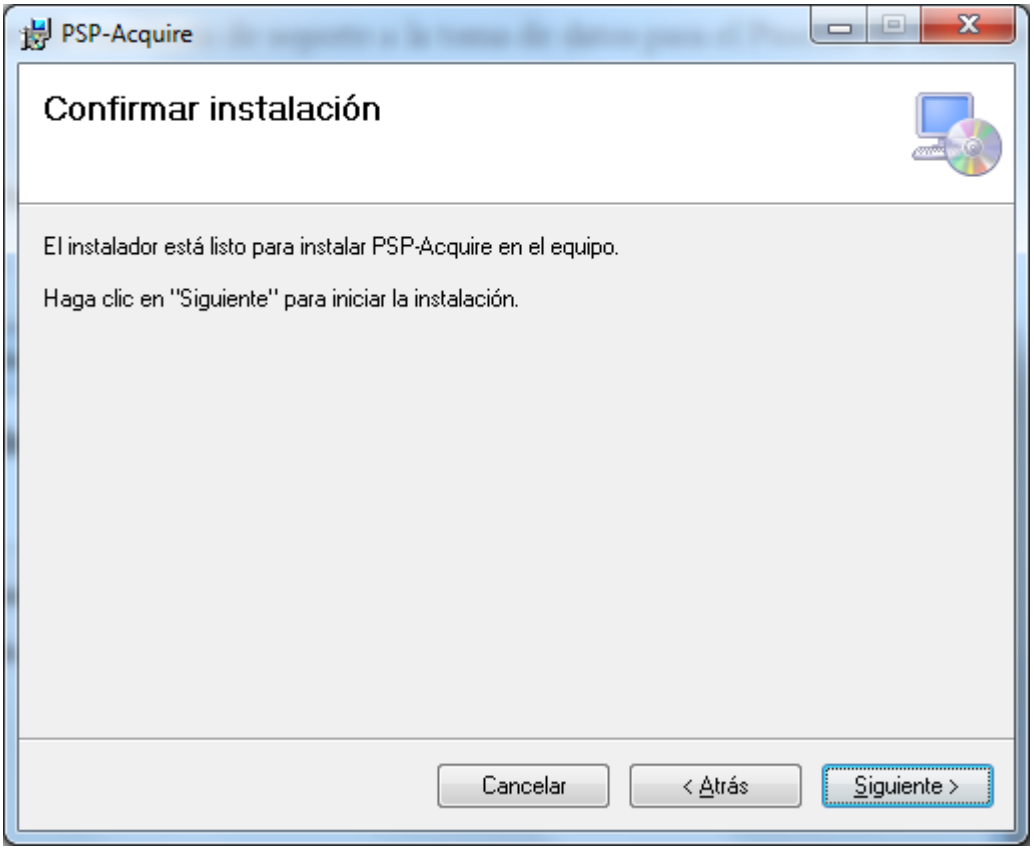


Ilustración 21: Instalación 3

En esta ventana el usuario debe esperar a que termine el proceso de instalación.

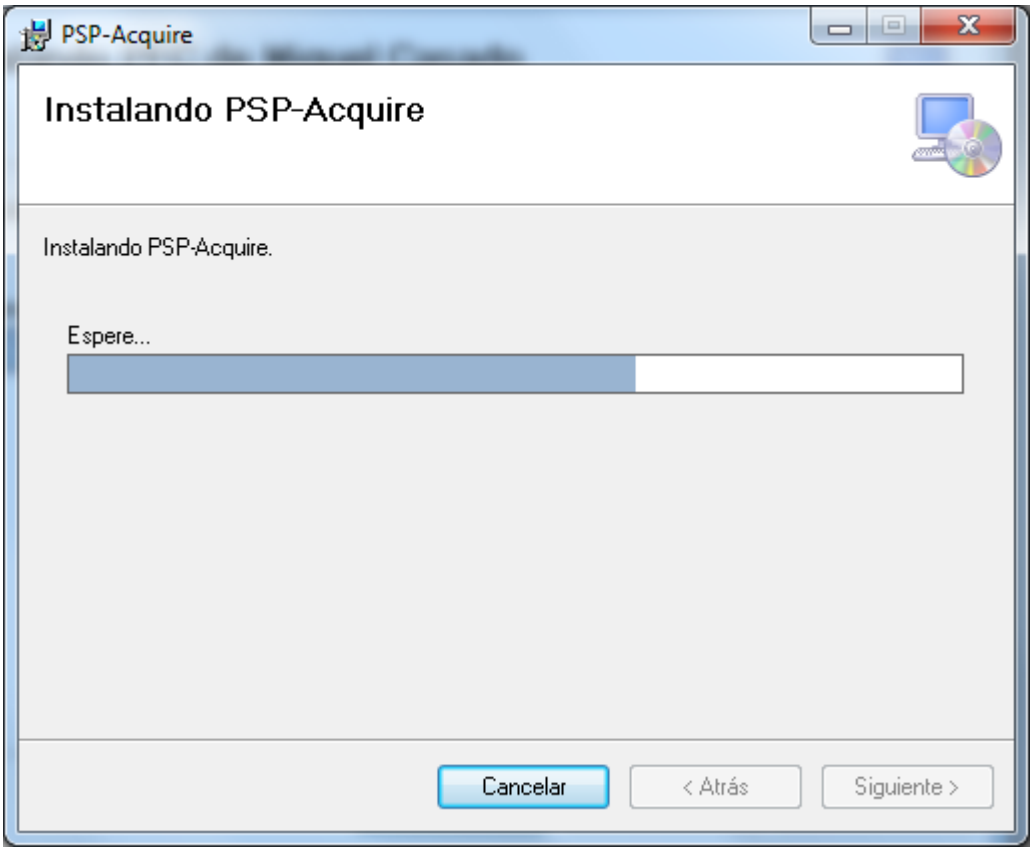


Ilustración 22: Instalación 4

Esta ventana muestra que la instalación ha sido completada correctamente.

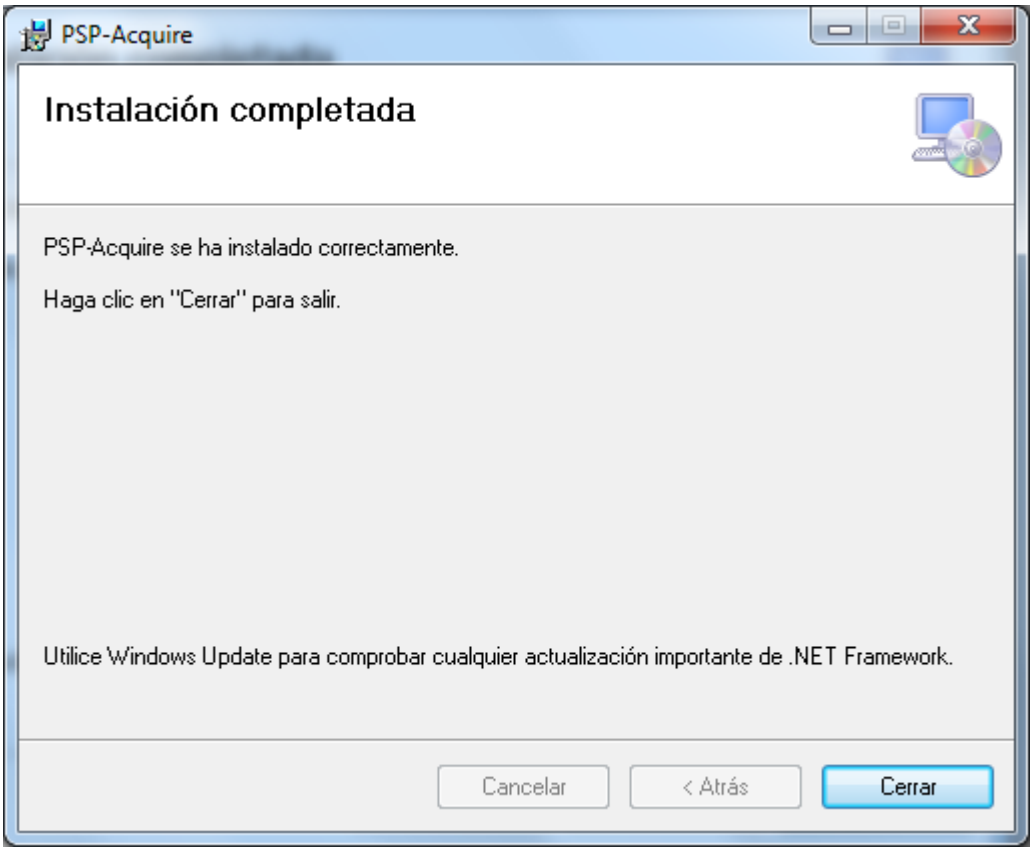


Ilustración 23: Instalación 5

Para la ejecución del programa se ha creado un acceso directo en el menú inicio.

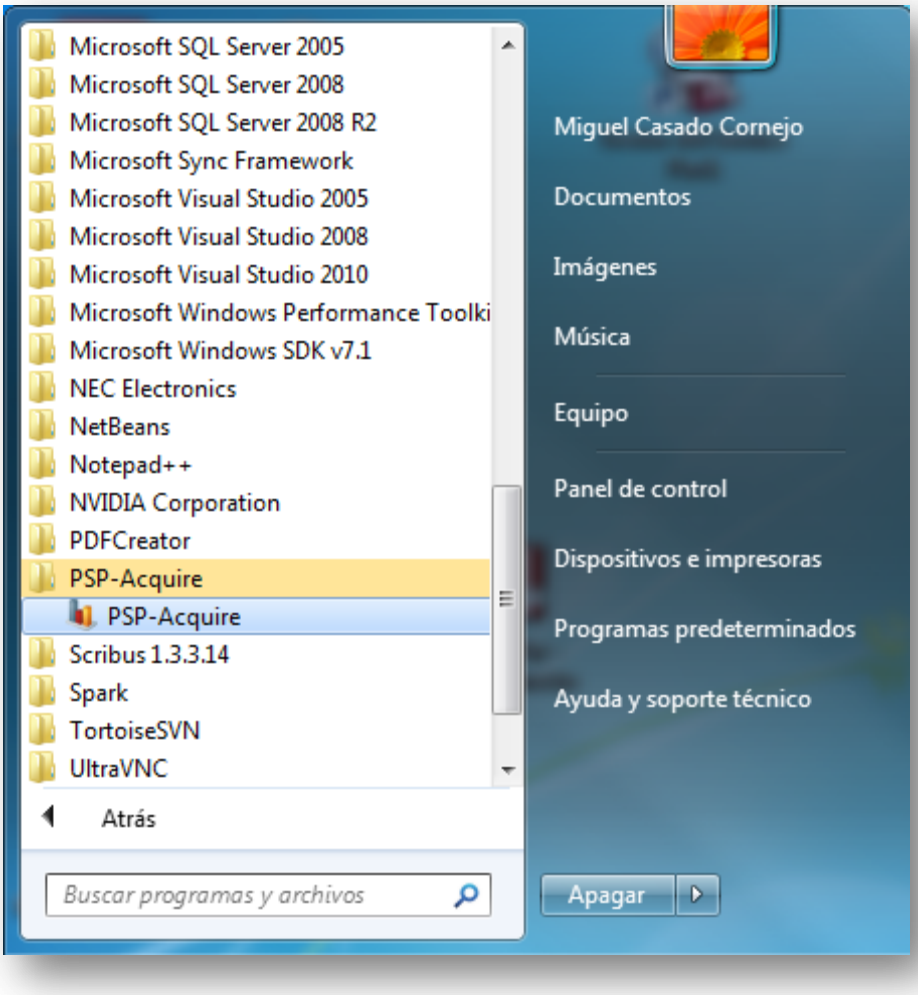


Ilustración 24: Instalación 6

Manual de usuario:

Validación en el sistema.

Para poder utilizar la aplicación el primer paso es validarse como un usuario válido. Justo después del proceso de instalación no existen usuarios activos por lo que se debe acceder a la aplicación con los datos siguientes:

- Nombre: admin
- Contraseña: 1234

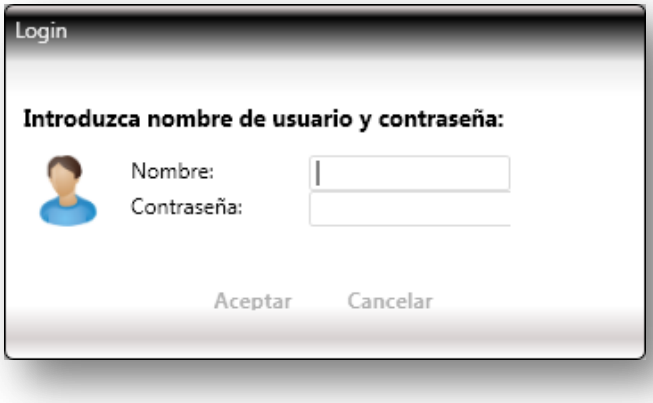


Ilustración 25: Ventana de Login

Para proceder a la validación del usuario debe pulsarse la tecla Enter o hacer clic en el botón Aceptar. Si el usuario existe y la contraseña es correcta el usuario pasa a estar validado en la aplicación, en caso contrario aparecerá un mensaje indicando el problema.

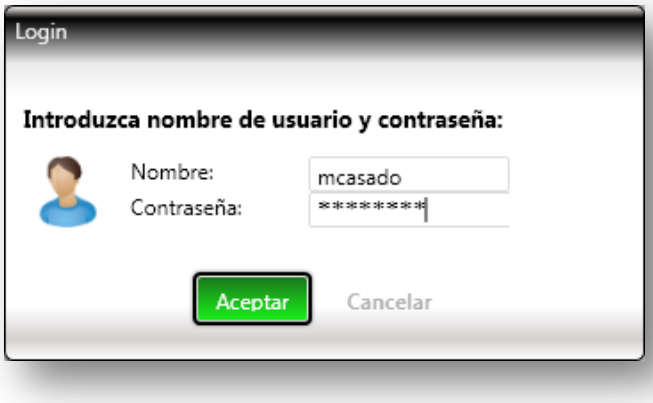


Ilustración 26: Ventana de Login. Aceptar

Si por el contrario el usuario no desea acceder todavía a la aplicación puede pulsar el botón cancelar y con ello se corta la ejecución del programa.

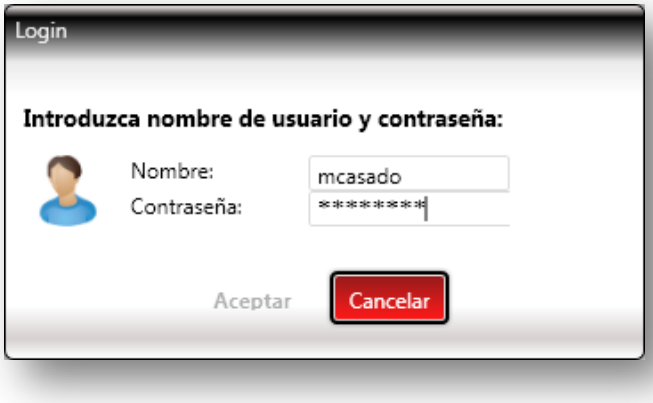


Ilustración 27: Ventana de Login. Cancelar

Una vez que el usuario se ha validado aparece el siguiente icono en la barra de notificaciones. Para mostrar la ventana principal de la aplicación sólo hay que posicionar el cursor del ratón encima del icono.

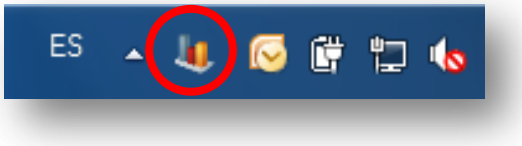


Ilustración 28: Icono de la aplicación

Cuando el usuario posiciona el cursor encima del cursor aparece la ventana principal de la aplicación en la siguiente posición dentro del escritorio del sistema.

Ilustración 29: Posición de la aplicación en el escritorio

Ventana principal

La ventana principal permite la navegación entre distintos subsistemas, así como la selección de actividad actual, programa actual y la toma de tiempos para una actividad concreta.

Existe la posibilidad de mantener la ventana "visible" durante toda la ejecución, ya que al retirar el cursor del botón de encima se oculta. Para mantener la ventana activa se debe hacer clic sobre el icono de la chincheta de la esquina superior derecha.

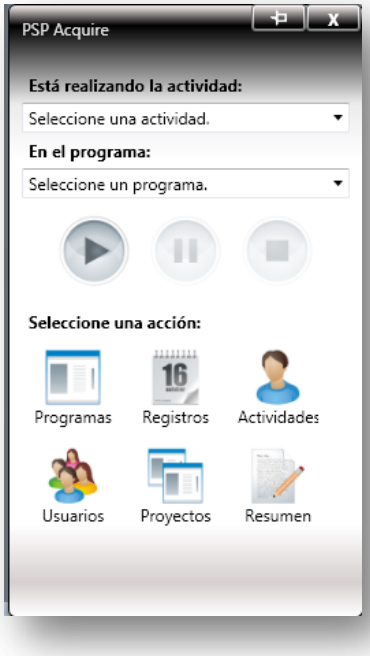


Ilustración 31: Ventana principal

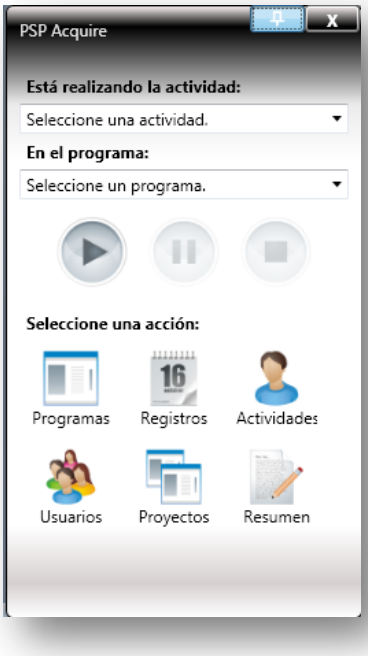


Ilustración 30: Ventana principal con chincheta

Toma de tiempos automática

Para comenzar con la toma de tiempos debe existir una actividad seleccionada, para ello se debe utilizar el desplegable correspondiente a la actividad actual. En dicho desplegable están tanto las actividades "básicas" del desarrollo de un programa como las actividades que el usuario introduzca en el sistema.

Ilustración 32: Selección de Actividad

Una vez seleccionada una actividad el usuario debe introducir el programa al que pertenece dicha actividad del mismo modo, es decir, seleccionando sobre un desplegable con los programas asociados al usuario activo. Existe la posibilidad de que la actividad no sea básica, es decir que no pertenezca a un programa, en ese caso se indicará deshabilitando el desplegable y cambiando la etiqueta mostrada a "Sin programa".

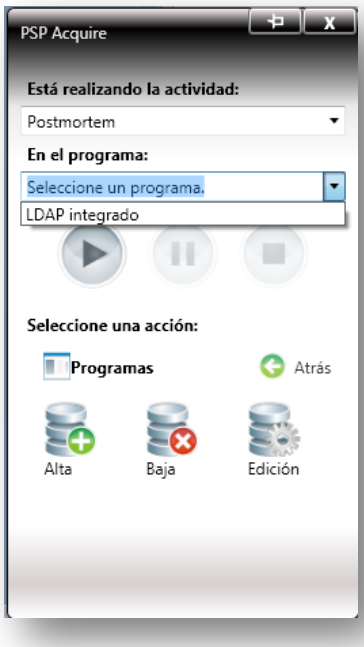


Ilustración 33: Selección de programa

Ilustración 34: Actividad sin programa

Una vez que el usuario tiene seleccionados tanto la actividad como el programa, en caso de que sea necesario, puede comenzar a tomar su tiempo pulsando el botón de Play. Una vez que se pulsa se comienza a tomar el tiempo y pasan a estar activos los botones de pausa y stop.

Ahora que el programa está tomando los tiempos el usuario debe realizar la actividad que ha marcado como activa.

Si el usuario necesita un descanso durante la realización de la actividad o ha tenido alguna interrupción debe pulsar el botón de pausa, para volver a presionar el botón de play cuando retome la actividad.

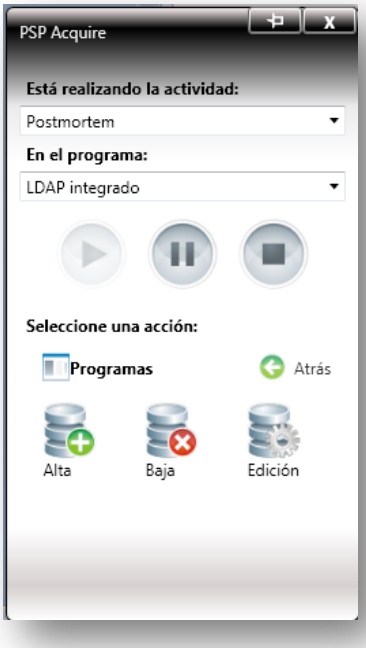


Ilustración 35: Comienzo de toma de tiempo

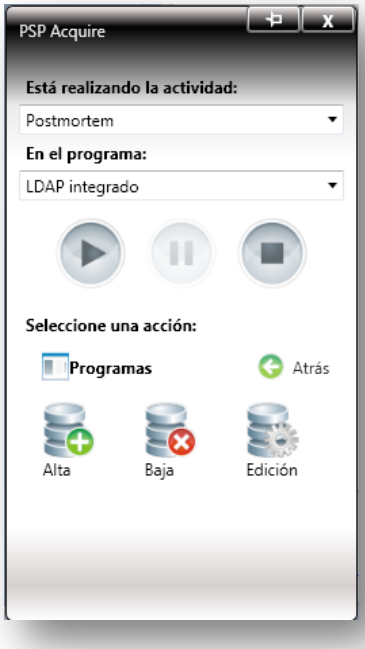


Ilustración 36: Pausa de toma de tiempo

Después del desarrollo de la actividad, ya sea con interrupciones o sin ellas, el usuario debe pulsar el botón de stop. Una vez pulsado se almacenará el tiempo dedicado a dicha actividad, pero antes se puede introducir un comentario para dicha actividad y durante el tiempo que se le ha dedicado.

El sistema muestra la siguiente pantalla para permitir la introducción de dicho comentario. Si el usuario elige sí aparece la ventana de introducción de comentarios, que una vez cerrada dará el comentario por concluido y almacenado en la base de datos.

Si el usuario elige no, se procede al almacenamiento del tiempo en el sistema sin la introducción de un comentario.

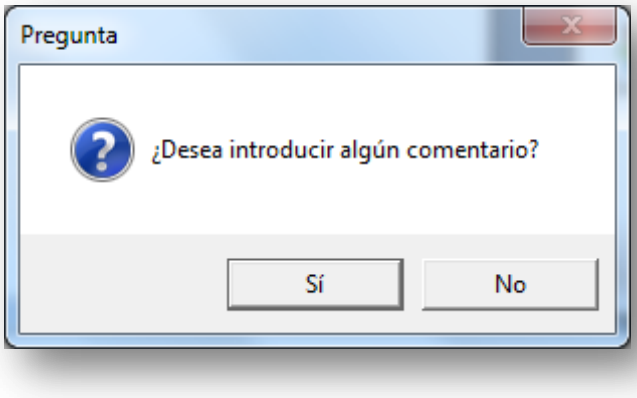


Ilustración 37: Inserción de comentario

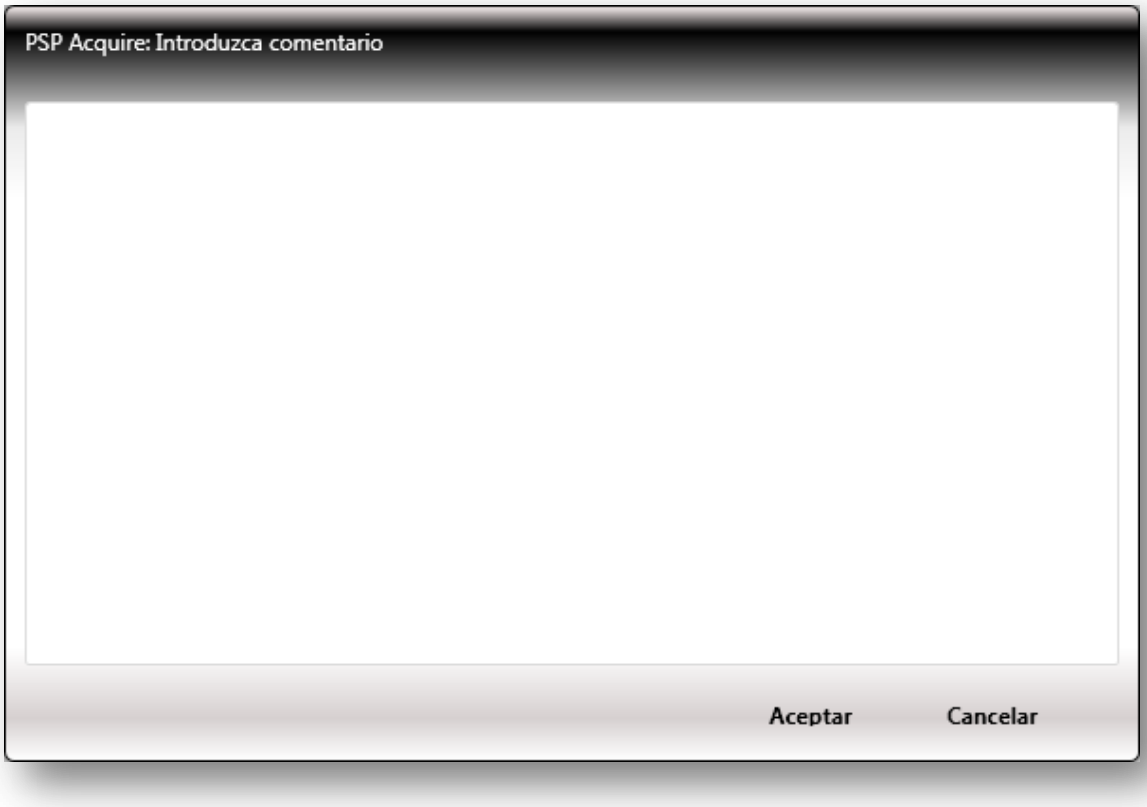


Ilustración 38: Página de inserción de comentario

Gestión de actividades.

El subsistema de gestión de actividades permite dar de alta una nueva actividad, borrar una actividad o modificar una existente.

Alta de actividad

El usuario debe introducir los datos de la actividad y presionar:

- Aceptar: si desea que se almacenen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se almacenen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se almacenen los datos, pero desea que la ventana siga abierta para introducir más datos.

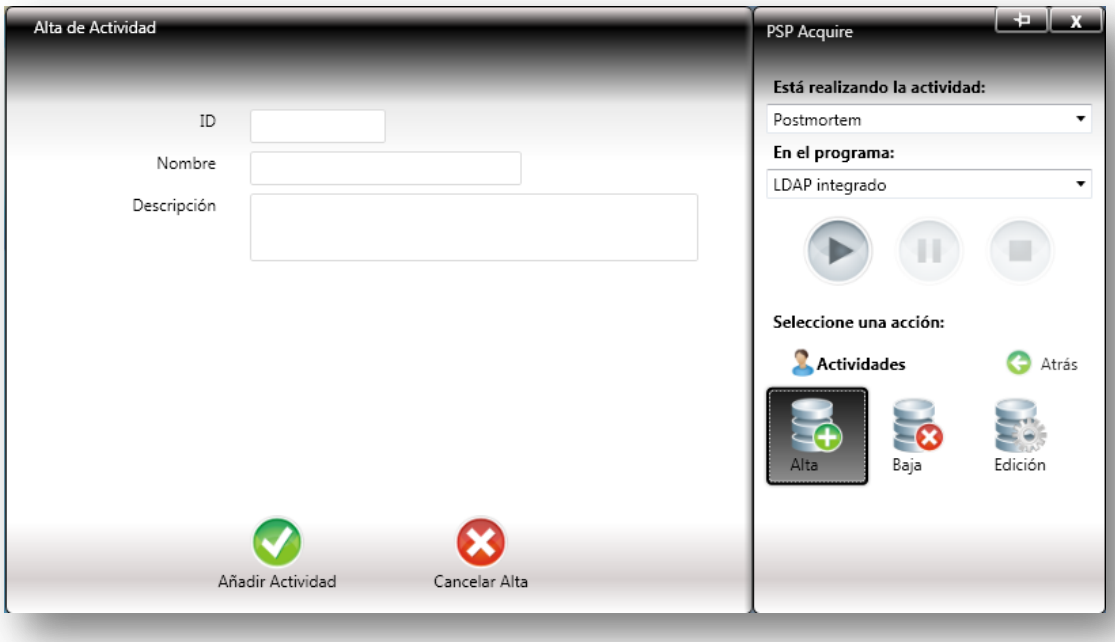


Ilustración 39: Alta de actividad

Baja de actividad

El usuario debe seleccionar una actividad y presionar:

- Aceptar: si desea que se borren los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se borren los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se borren los datos, pero desea que la ventana siga abierta para borrar más datos.

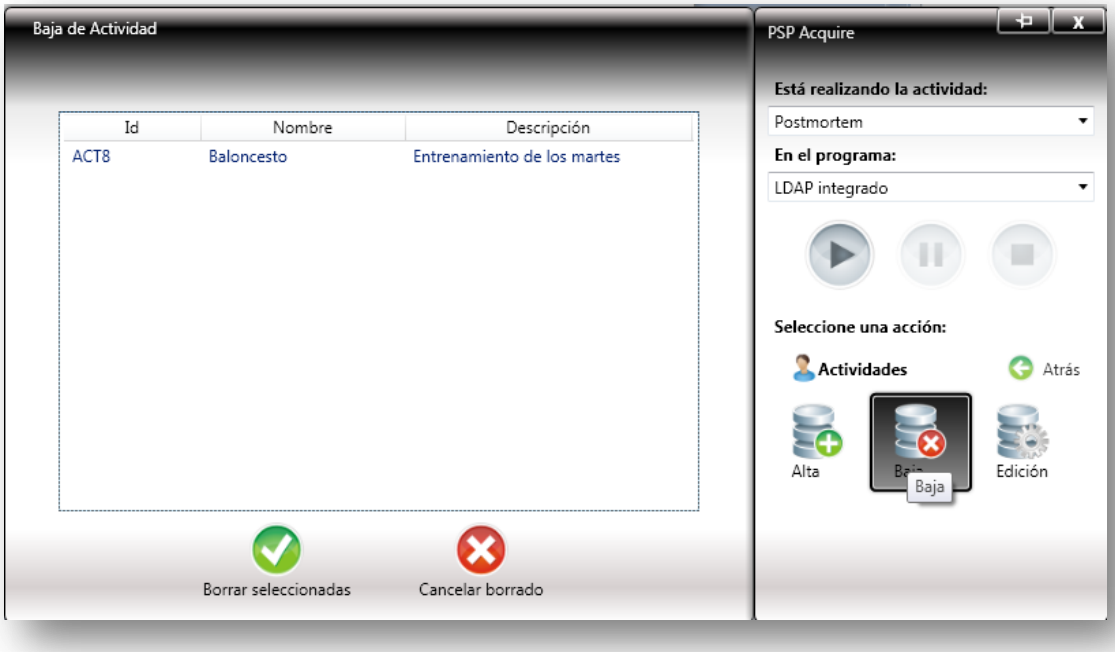


Ilustración 40: Baja de Actividad

Edición de actividad

El usuario debe seleccionar una actividad, editar sus datos y presionar:

- Aceptar: si desea que se editen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se editen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se editen los datos, pero desea que la ventana siga abierta para editar más datos.

Ilustración 41: Edición de actividad

Alta de programa

El usuario debe introducir los datos del programa y presionar:

- Aceptar: si desea que se almacenen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se almacenen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se almacenen los datos, pero desea que la ventana siga abierta para introducir más datos.

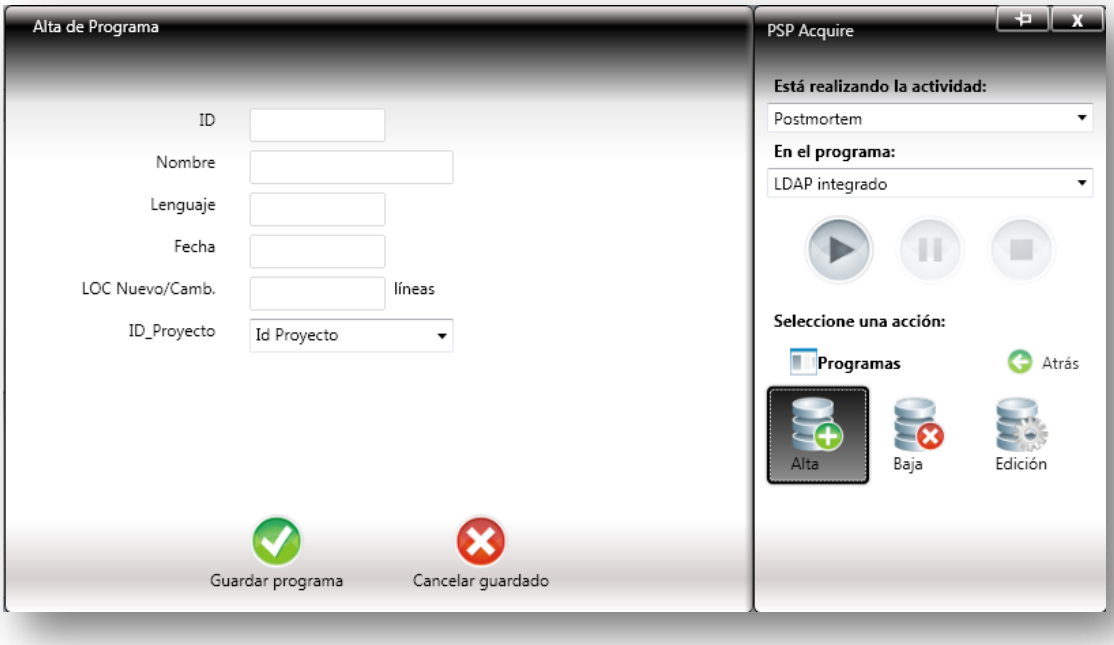


Ilustración 42: Alta de programa

Baja de programa

El usuario debe seleccionar un programa y presionar:

- Aceptar: si desea que se borren los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se borren los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se borren los datos, pero desea que la ventana siga abierta para borrar más datos.

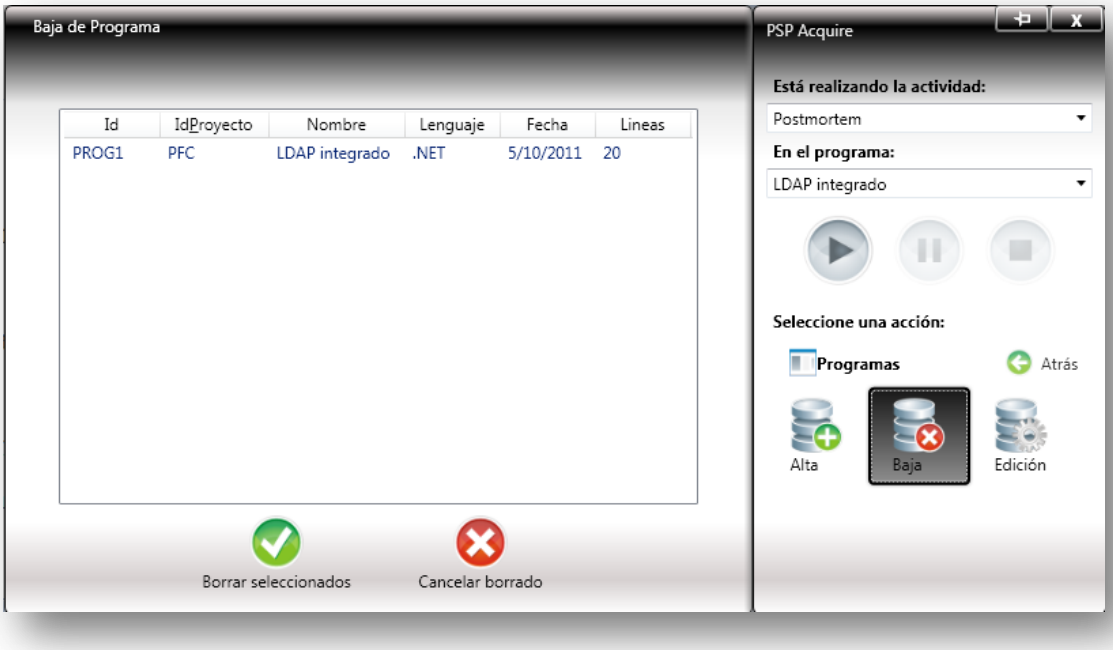


Ilustración 43: Baja de programa

Edición de programa

El usuario debe seleccionar un programa, editar sus datos y presionar:

- Aceptar: si desea que se editen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se editen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se editen los datos, pero desea que la ventana siga abierta para editar más datos.

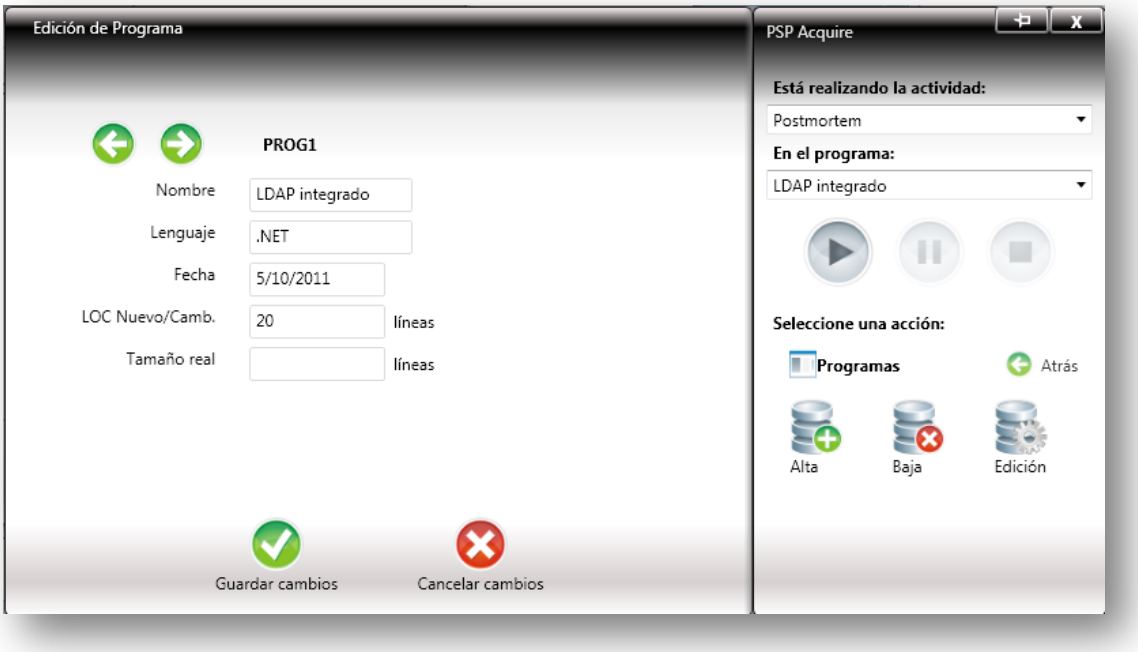
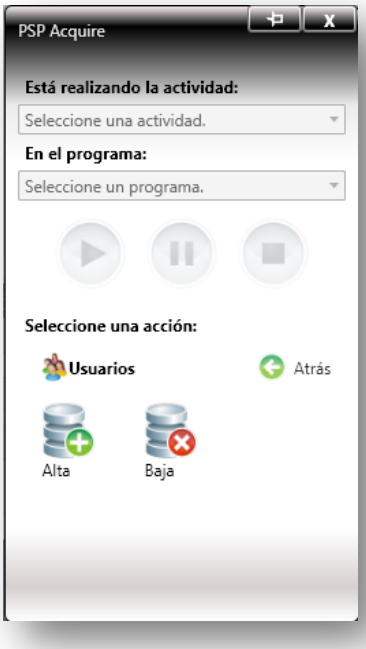


Ilustración 44: Edición de programa

Alta de usuario



El usuario debe validarse en el sistema como el usuario admin. Puesto que es el único usuario que puede dar de alta o baja a usuarios.

Una vez validado como administrador las opciones a elegir son el alta o la baja de usuarios.

El usuario debe introducir los datos del usuario y presionar:

- Aceptar: si desea que se almacenen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se almacenen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se almacenen los datos, pero desea que la ventana siga abierta para introducir más datos.

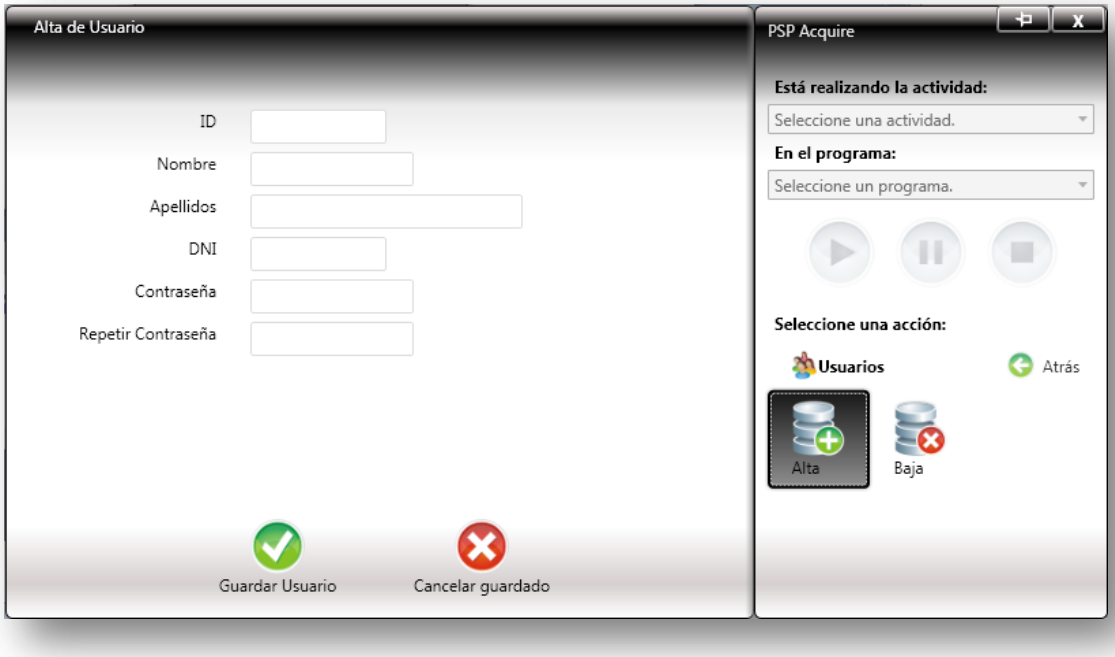


Ilustración 45: Alta de Usuario

Baja de Usuario

El usuario debe seleccionar un usuario y presionar:

- Aceptar: si desea que se borren los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se borren los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se borren los datos, pero desea que la ventana siga abierta para borrar más datos.

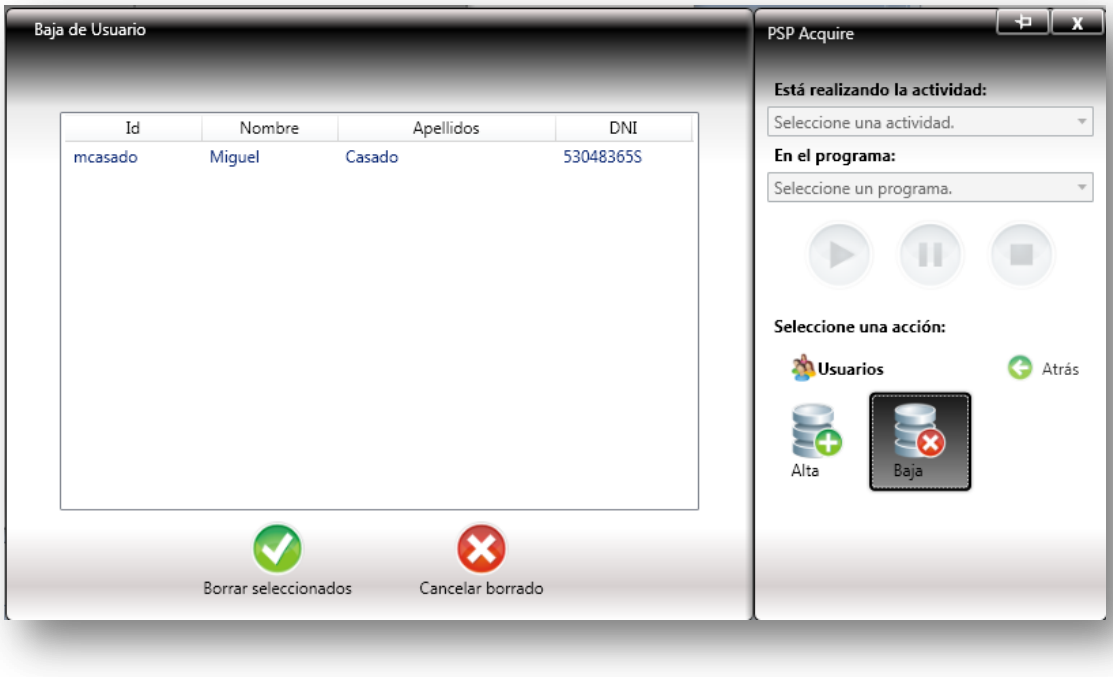


Ilustración 46: Baja de Usuario

Edición de usuario

El usuario editar sus datos e introducir su contraseña y presionar:

- Aceptar: si desea que se editen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se editen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se editen los datos, pero desea que la ventana siga abierta para editar más datos.

Ilustración 47: Edición de usuario

Cambio de contraseña

El usuario debe introducir los datos y presionar:

- Aceptar: si desea que se editen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se editen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se editen los datos, pero desea que la ventana siga abierta para editar más datos.

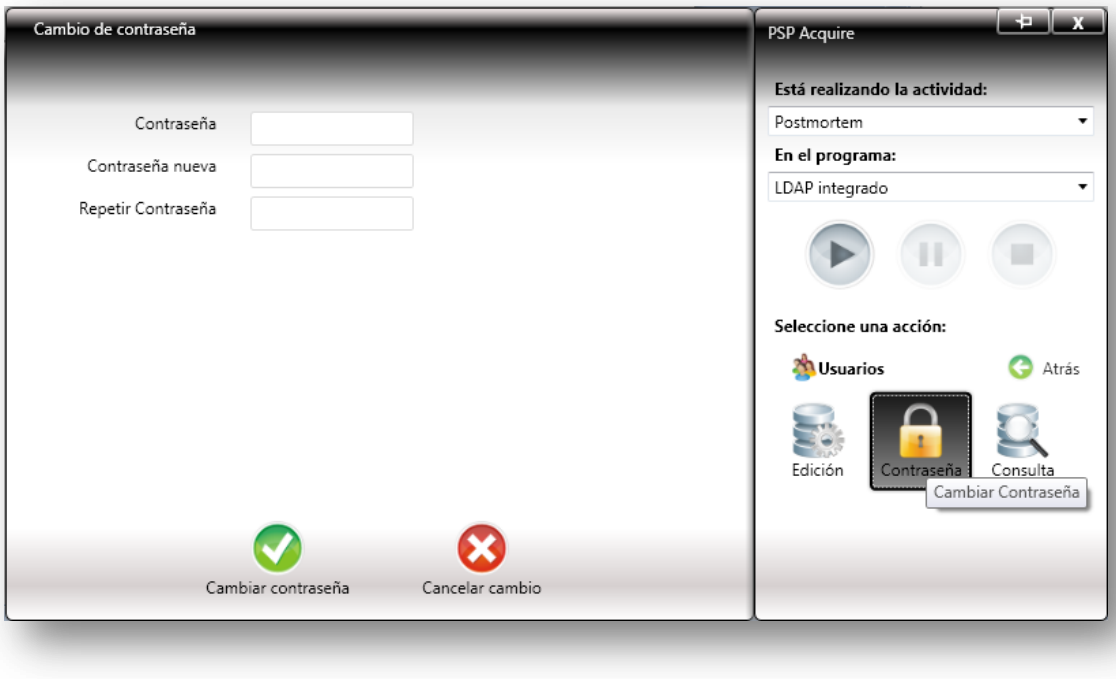


Ilustración 48: Cambio de contraseña

Consulta de usuario

El usuario puede navegar por los datos de los usuarios del sistema, pero no puede editar sus datos:

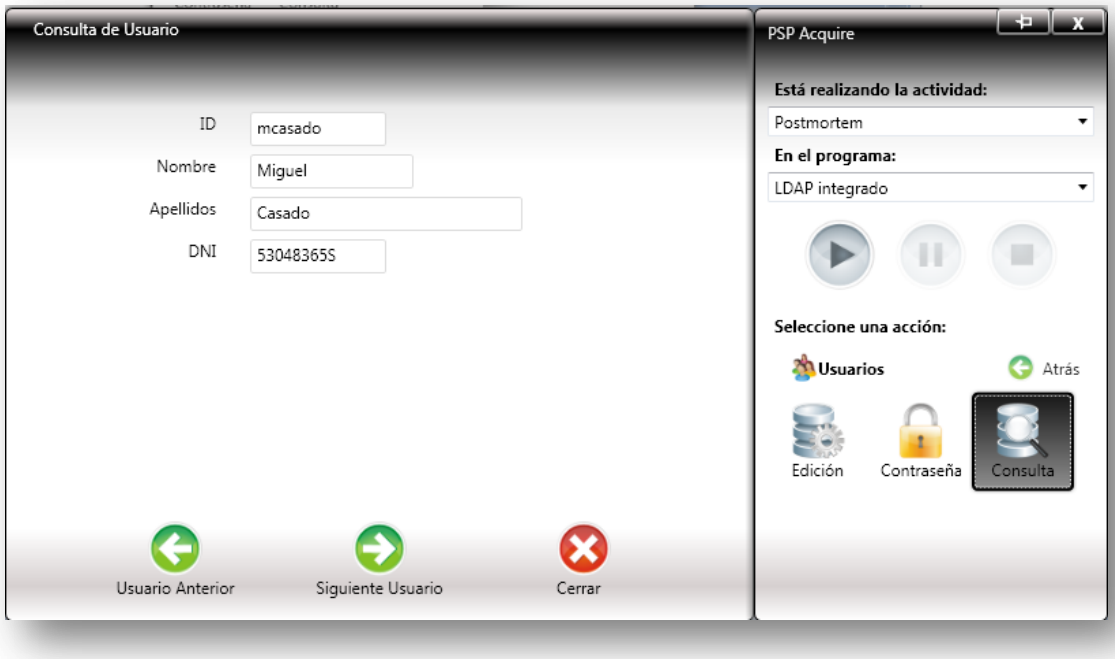


Ilustración 49: Consulta de usuario

Alta de proyecto

El usuario debe introducir los datos del proyecto y presionar:

- Aceptar: si desea que se almacenen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se almacenen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se almacenen los datos, pero desea que la ventana siga abierta para introducir más datos.

Ilustración 50: Alta de proyecto

Baja de proyecto

El usuario debe seleccionar un proyecto y presionar:

- Aceptar: si desea que se borren los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se borren los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se borren los datos, pero desea que la ventana siga abierta para borrar más datos.

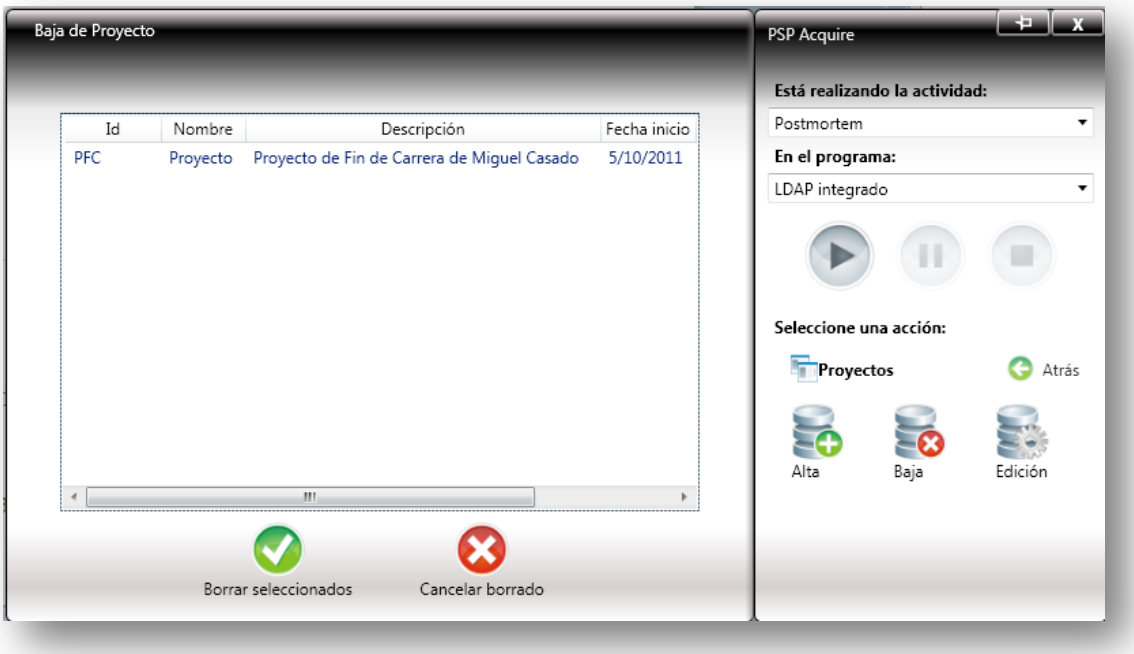


Ilustración 51: Baja de Proyecto

Edición de proyecto

El usuario debe seleccionar un proyecto, editar sus datos y presionar:

- Aceptar: si desea que se editen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se editen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se editen los datos, pero desea que la ventana siga abierta para editar más datos.

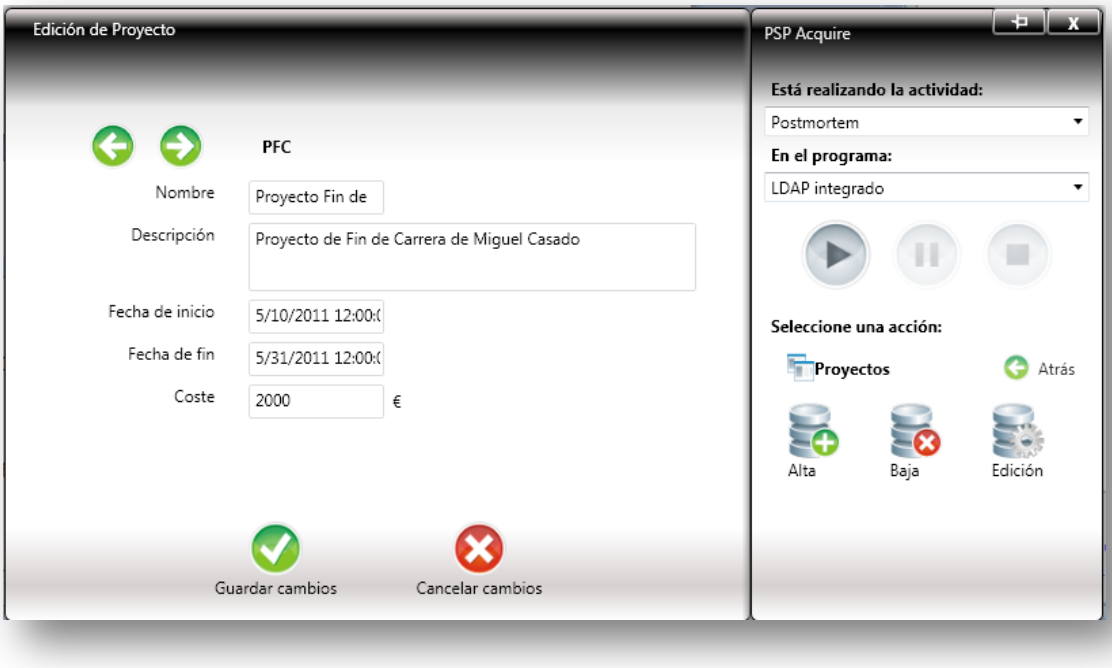


Ilustración 52: Edición de proyecto

Alta de defecto

El usuario debe introducir los datos del defecto y presionar:

- Aceptar: si desea que se almacenen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se almacenen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se almacenen los datos, pero desea que la ventana siga abierta para introducir más datos.

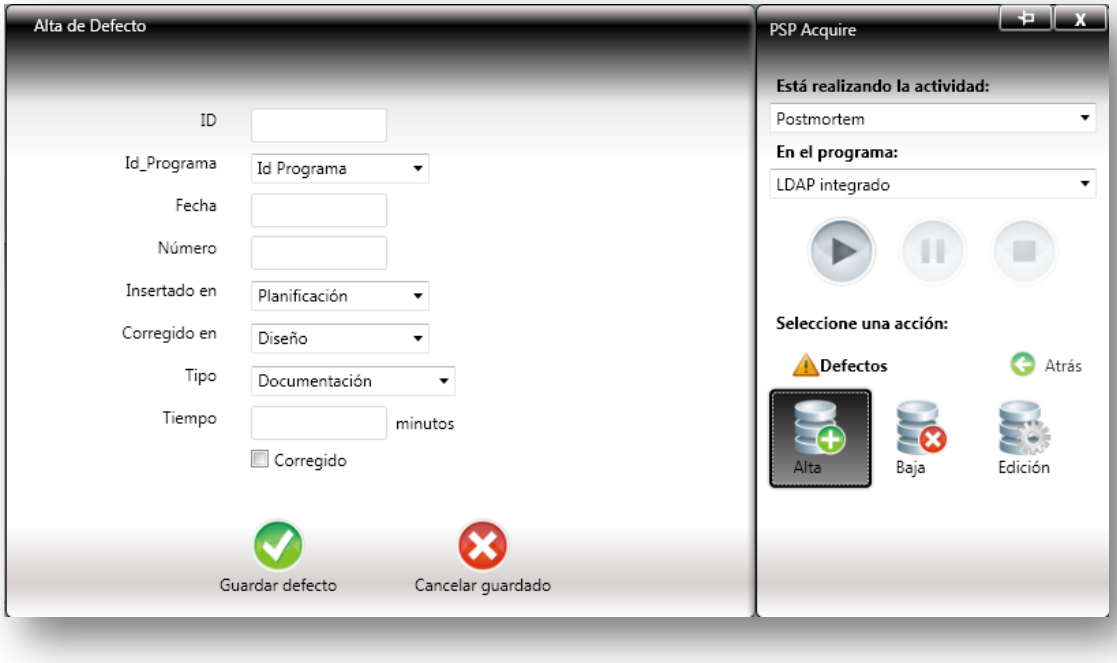


Ilustración 53: Alta de defecto

Baja de defecto

El usuario debe seleccionar un defecto y presionar:

- Aceptar: si desea que se borren los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se borren los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se borren los datos, pero desea que la ventana siga abierta para borrar más datos.

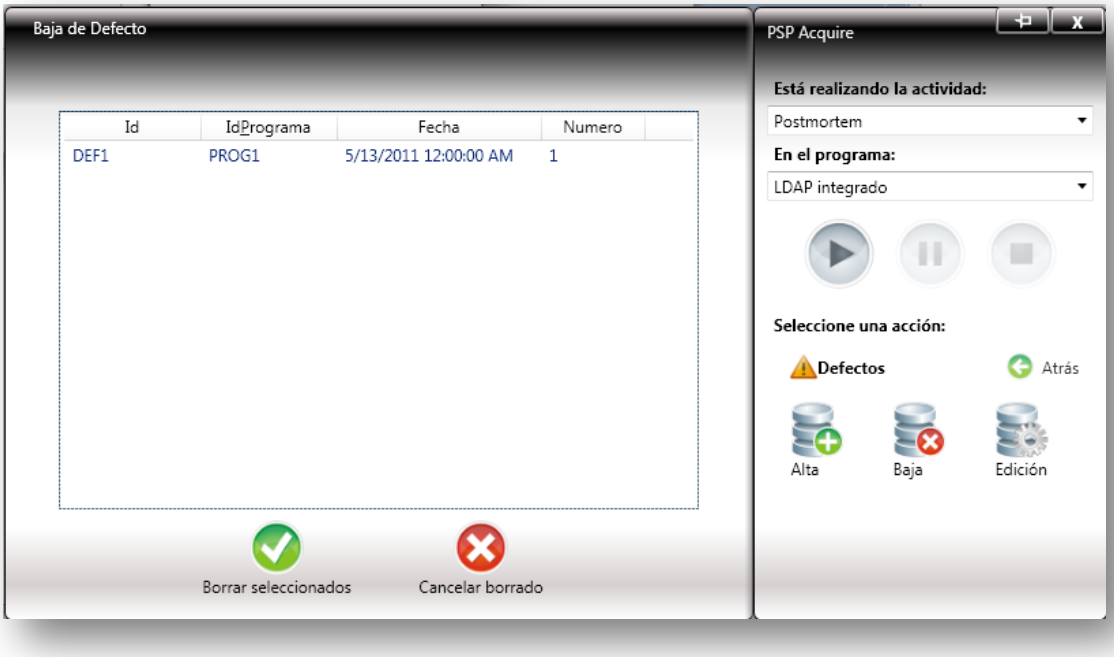


Ilustración 54: Baja de defecto

Edición de defecto

El usuario debe seleccionar un defecto, editar sus datos y presionar:

- Aceptar: si desea que se editen los datos y se cierre la ventana de introducción de datos.
- Cancelar: si desea que NO se editen los datos y se cierre la ventana de introducción de datos.
- Aplicar: si desea que se editen los datos, pero desea que la ventana siga abierta para editar más datos.

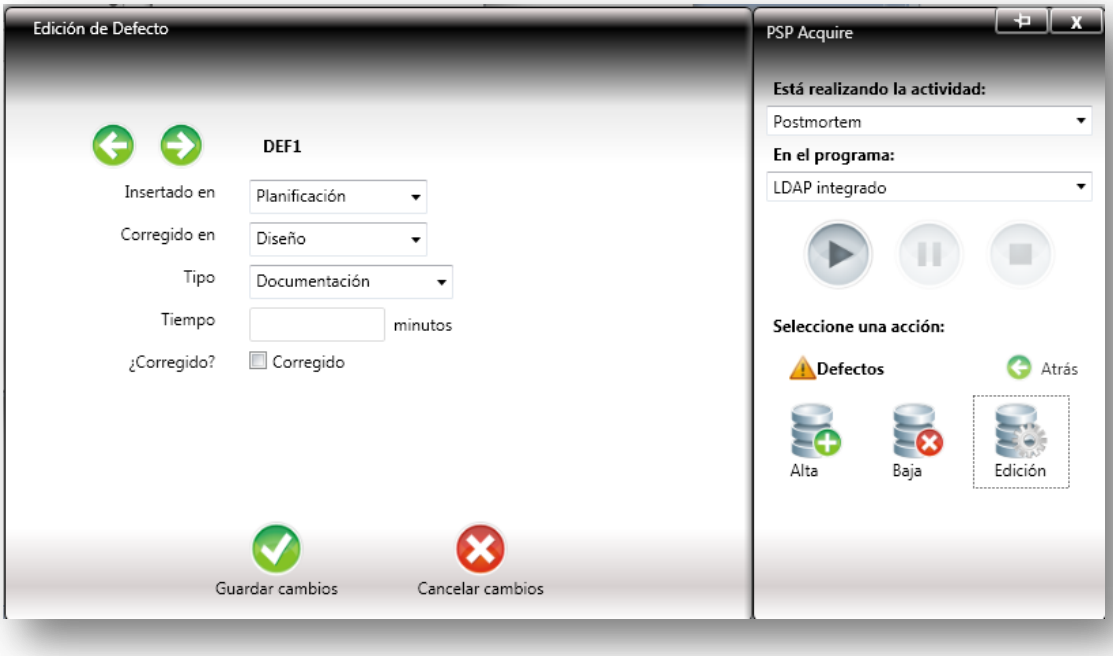


Ilustración 55: Edición de defecto

Resumen semanal de actividades

El usuario debe seleccionar tanto un proyecto como una semana. El sistema automáticamente genera los datos a mostrar representando en columnas las actividades y mostrando una fila por cada día de la semana.

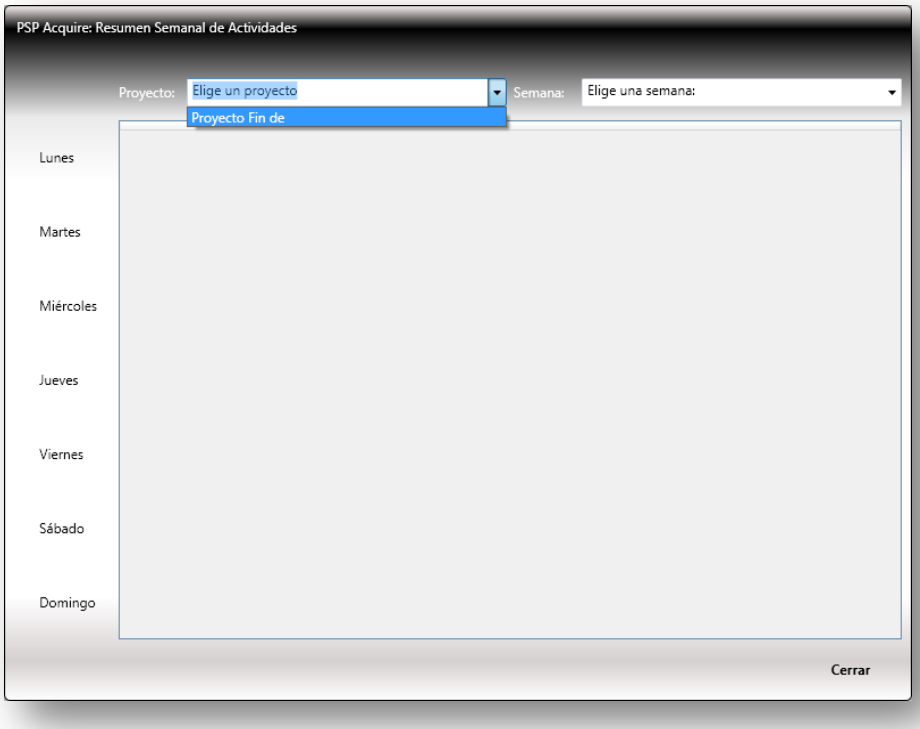


Ilustración 57: RSA 1

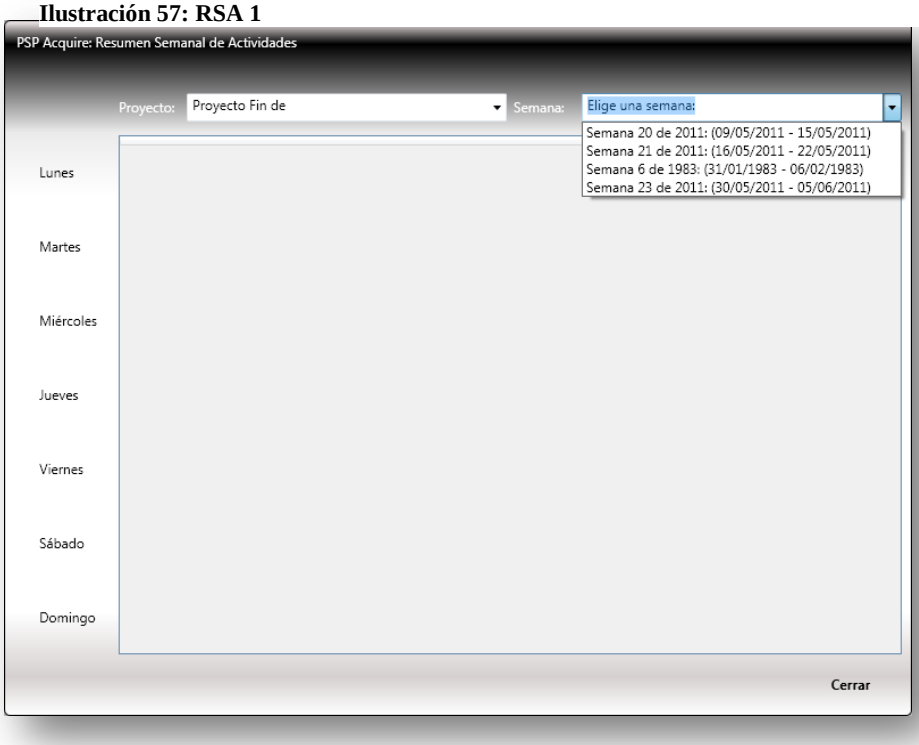
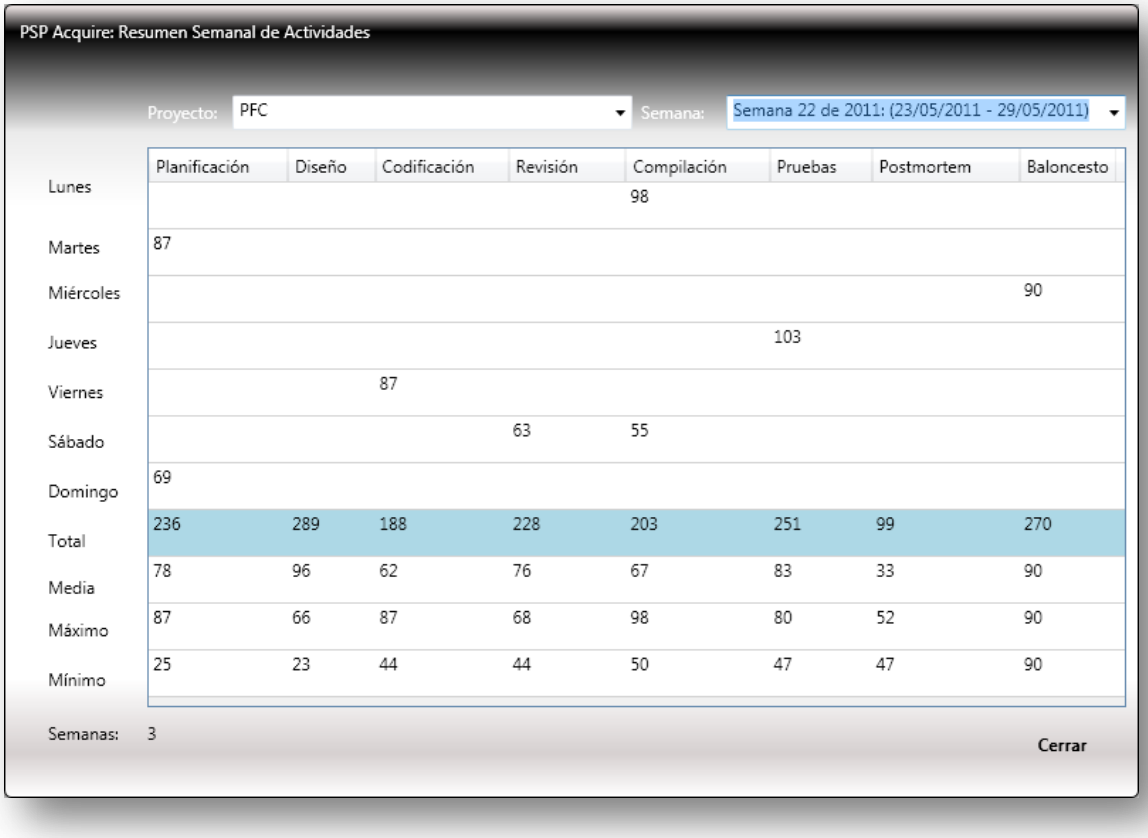


Ilustración 56: RSA 2



	Planificación	Diseño	Codificación	Revisión	Compilación	Pruebas	Postmortem	Baloncesto
Lunes					98			
Martes	87							
Miércoles								90
Jueves						103		
Viernes			87					
Sábado				63	55			
Domingo	69							
Total	236	289	188	228	203	251	99	270
Media	78	96	62	76	67	83	33	90
Máximo	87	66	87	68	98	80	52	90
Mínimo	25	23	44	44	50	47	47	90

Ilustración 58: RSA 3

Conclusiones y líneas futuras:

Conclusiones:

Durante el desarrollo de esta memoria se ha detallado un breve resumen del trabajo realizado para la conclusión satisfactoria del proyecto, realizando un ciclo de vida completo, puesto que se ha realizado desde el análisis hasta la implementación e implantación del producto.

Dicho ciclo de vida ha permitido adquirir experiencias dentro del desarrollo de un proyecto, realizando todas sus fases.

Una vez que se han realizado con éxito todas las fases se consigue un producto final, que en este caso ha sido una aplicación para la toma de tiempos automatizada.

La aplicación desarrollada se puede considerar un prototipo avanzado, puesto que muestra la funcionalidad para la que fue creada, pero quedando muchos de los requisitos de capacidad sin realizar. Aún así puede ser utilizado por estudiantes o estudiantes recién titulados para conseguir administrar su tiempo de forma más eficiente.

El paso por todas las fases del desarrollo de un proyecto permite asentar todos los conocimientos adquiridos durante el estudio de la carrera, por lo que bajo mi opinión de recién titulado es una parte fundamental del desarrollo completo de los estudios, ya que permite afrontar con un poco más de confianza y seguridad el comienzo de la vida laboral.

Líneas futuras de investigación:

Al igual que la tendencia global del PSP se ha adaptado hacia el desarrollo de nuevas planificaciones como el TSP (Team Software Proccess), la evolución lógica del sistema sería avanzar en la misma dirección y crear un módulo para equipos de desarrollos.

El TSP mejora significativamente PSP, al tener en cuenta que los desarrollos no son exclusivos de un único usuario, sino de un grupo de desarrollo, asignando a cada uno roles y aportando conocimiento al diseño del plan de proyecto.

Referencias:

- Introducción al proceso de software Personal. Watts S. Humpfrey. Pearson Educación.

- **Patrones de Diseño. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** Pearson Educación.

- Wikipedia. http://www.wikipedia.es.

- Software Engineering Institute (SEI). http://www.sei.cmu.edu.

- Msdn. **msdn**.microsoft.com/es-es/default.aspx